Dr. Dobb's Journal of

#116 JUNE 1986
$2.95 (3.95 CANADA)

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

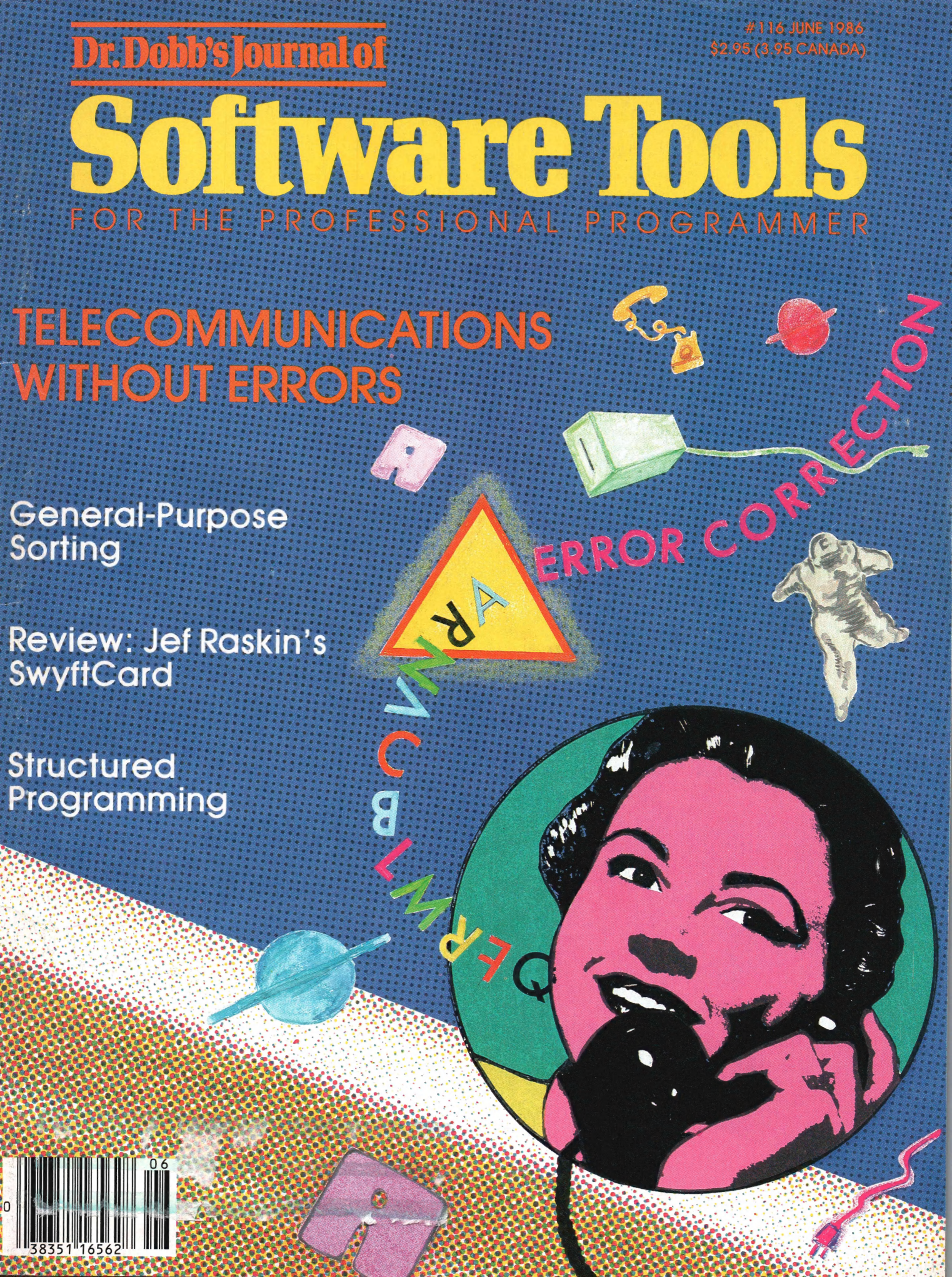
TELECOMMUNICATIONS WITHOUT ERRORS

ERROR CORRECTION

General-Purpose Sorting

Review: Jef Raskin's SwyftCard

Structured Programming



0 38351 16562 06

This is the Modula-2 compiler *everybody's* been waiting for...

LOGITECH MODULA-2/86

Now, you can cross the bridge to Modula-2 with ease.

This is Modula-2 at its *absolute* best. It's a fully integrated development environment that takes into account what you need as a programmer. Without leaving the Editor, you can call the compiler, linker and utilities.

With Logitech's Modula-2, you'll have the ability to edit several files at once, comparing, window to window, various code modules. You can even move from window to window compiling, linking, debugging and running.

The compiler has the kind of power and room to breathe that you really need in today's complex applications. It is as *easy* to use as Turbo Pascal, without your programs being limited to 64K of code.

At your command will be the libraries of modules that make Modula-2 a programmer's dream. It has essentially the same structure as Pascal with the major addition of a library organization of code modules that allow you to put together programs on a solid, block-by-block, foundation of proven code.

Whether you're working with a module of your own making, or one of the many in our library, you'll find the system by which each module is identified, described and stored an organizational masterpiece. And that's at the heart of Modula-2.

Underneath the sophisticated system is a Modula-2 compiler that is the result of years of development and proven use in industry. We run on the Vax*, and we run on the IBM PC. And the code is portable–from one to the other.

Best of all . . . you can have it right now!

Logitech Modula-2/86 $89 Complete with Editor, Run Time System, Linker, Cursor-positioning debugger, 8087 Software Emulation, BCD module, Logitech's extended library, Utility to generate standard .EXE files, Turbo Pascal (and standard Pascal, too) to Modula-2 translator *(included without charge until 8/1/86)*, and much, much more!

Logitech Modula-2/86 with 8087 support $129 Even if you haven't yet gotten an 8087 co-processor, you can still use this version.

Logitech Modula-2/86 Plus $189 For machines with 512K or more. Takes advantage of the larger memory to increase compilation speed by 50%! Supports 80186 and 80286 as well as 8086 and 8088. Includes 8087 and 80287 support, too.

Window Package $49 Now you can build true windowing into your Modula-2/86 code with ease, too. Very powerful and very full, yet only 15K in size. Features virtual screens, color support, overlapping windows and a variety of borders.

Run Time Debugger (source level) $69 Much more powerful than just a symbolic RTD. Display source code, data, procedure call chain and raw memory. Set break points, assign values to variables, pinpoint and identify bugs in your source. The ultimate professional's tool!

Utilities Package $49 Features a post-mortem debugger for static debugging. If a program you've written crashes at run time, the situation is frozen, and you can pinpoint, in source, the cause of the error and the data at that moment. Also includes a disassembler, a cross reference utility and a "version" utility that allows conditional compilation.

Make Utility $29 Automatically selects modules affected by code changes for quick and minimal re-compilation and relinking. Even figures out dependencies for you.

Library Sources $99 Source code for our major library modules is now available–for customization or exemplification.

ROM Package If you need to produce rommable code, call our 800 number for further information on this package.

To place an order call our special toll free number

800-231-7717

in California

800-552-8885

Special offer until 8/1/86! **$89**

includes

Free! $49.95 value Turbo Pascal translator!

Now, you can take your library with you!

Yes, I'd like to take the next logical step in programming.

Please send my copy of Logitech Modula-2/86 to the following address:

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____

State _____ Zip _____ Phone (_____) _____

Here's the configuration I'd like:

☐ Logitech Modula-2/86 $89

☐ Logitech Modula-2/86 $129 with 8087 support

☐ Logitech Modula-2/86 Plus $189

Please add $6.50 for shipping and handling.

And include the indicated items:

☐ Window Package $49

☐ Run Time Debugger $69 (source level)

☐ Utilities Package $49

☐ Make Utility $29

☐ Library Sources $99

Total enclosed $ _____

(California residents, please add applicable sales tax)

LOGITECH

LOGITECH, Inc.
805 Veterans Boulevard
Redwood City, California 94063
Telephone (415) 365-9852

For European pricing, please contact:

LOGITECH SA
Box 32, CH-1143 Apples, Switzerland
Telephone 41 (21) 774545

Please call our 800 line for: ☐ Information on our *VAX version ☐ Site License and University Discounts ☐ Dealer and Distributor information

*Turbo Pascal is a registered trademark of Borland International

PC Magazine
Product of the Year

Macro Assembler

The quickest. Bar none.

Our Macro Assembler has long been the most complete package on the market. Now it's also the fastest. Three times faster than before. And faster than anyone else. Period.

Of course, it's still the most powerful assembler on the market. It supports the standard 8086/8087 opcodes. And the new 186/286/287 instruction set. So you can make the most of the new machines.

Debugging is quicker, too. Thanks to our interactive symbolic debugger, SYMDEB. Now you can refer to variables and source code instead of getting lost in hex dumps. And this debugger also works with Microsoft languages like C, FORTRAN and Pascal. So now you can set breakpoints and trace execution—using source code for reference.

Cut your development time dramatically. Microsoft Macro Assembler's Symbolic Debug utility lets you debug your Macro Assembler programs, or debug your Microsoft C, FORTRAN or Pascal programs using your original source code or the resulting disassembly. For example, you can set breakpoints on line numbers and observe the contents of variables or expressions.

SYMDEB is just part of our complete set of utilities. Tools that make programming as fast as it should be. There are the linker and library managers you'd expect. Plus a new version of MAKE, our maintenance utility, with improvements like macro expansions and inference rules.

We've also revised the manuals. Our new Macro Assembler has a lot to offer, so we added more examples. Now our manuals are not only thorough, they're clearer than ever before.

For quick development and assembly, the choice is obvious. Microsoft. There's nobody faster.

Microsoft® Macro Assembler Version 4.0 for MS-DOS®

Macro Assembler
- Fastest macro assembler for MS-DOS computers.
- Supports the 8086/8087/8088 and the 186/286/287.
- Define macros.
- Conditional assembly.
- Optional case sensitivity for symbols.
- 100% upward compatibility from earlier versions of both the Microsoft and IBM® Macro Assemblers.

Interactive Symbolic Debug Utility
- Source level debugger for programs written in Microsoft Macro Assembler, C Compiler, FORTRAN, and Pascal.
- Screen swapping helps debug highly visual applications.
- Set breakpoints on line numbers and symbols.
- Single step to follow program execution.
- Disassemble object code.
- Display and modify values.
- Full I/O redirection.

Program Maintenance Utility
- Rebuilds your applications after your source files have changed.
- Similar to UNIX™ MAKE utility.
- Supports macro definitions and inference rules.

Library Manager
- Create, organize and maintain your object module libraries created with Microsoft languages.
- Set page size from 16 to 32678, to create compact and granular libraries.

Object Code Linker
- Simple overlaying linker combines relocatable object modules created using Microsoft languages into a single program.
- Load Map generation.
- Specify from 1 to 1024 segments.

Cross-Reference Utility
- Creates a cross-reference listing of the definitions and locations of all symbols used in an assembly language program, which makes debugging programs easier.

Microsoft EXE File Compression Utility
- Packs EXE files for smaller size on disk and faster loading at execution time.

Microsoft EXE File Header Utility
- Display and modify EXE file header, allowing you to tune the stack size and initial memory allocation.

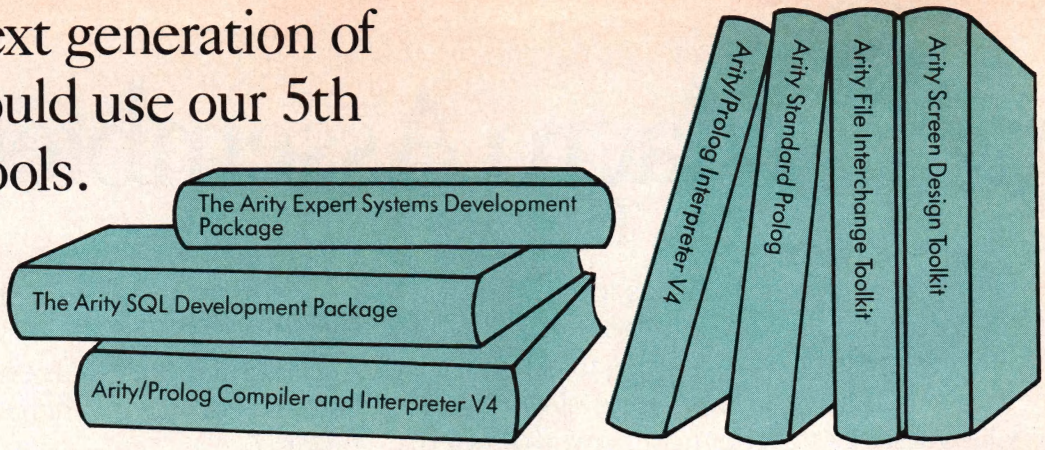
For the name of your nearest Microsoft dealer call (800) 426-9400. In Washington State and Alaska, call (206) 882-8088. In Canada, call (416) 673-7638.

Microsoft

The High Performance Software™

Microsoft and MS-DOS are registered trademarks and The High Performance Software is a trademark of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation. UNIX is a trademark of AT&T Bell Laboratories.

Why your next generation of products should use our 5th generation tools.



Arity's integrated family of programming tools allows you to combine software written in Arity/Prolog, the best of the fifth generation languages, with Arity SQL, the best of the fourth generation languages, and with conventional third generation languages such as C or assembly language to build your smarter application.

You can use Arity/Prolog to build expert systems using the Arity Expert Systems Development Package. Or to build natural language frontends. Or to build intelligent information management systems. Arity/Prolog lets you build advanced technology into your vertical applications package.

And more...

That's not the whole story. Arity's products are all designed to be fast, powerful, serious. Each of our products contains unexpected bonuses. Such as a one gigabyte virtual database integrated into Arity/Prolog. The most powerful of its kind on a PC.

Quality first. Then price.

In order to be the best, we had to prove it to our customers. Our tradition of quality software design is reflected in every product we sell. Quality first. Then price. And we always provide the best in customer support.

Our products are not copy protected. We do not charge royalties. We offer generous educational and quantity discounts. And we have a 30 day money back guarantee.

Try us to know that we keep our promise on commitment to quality and reliability. Try us by using our electronic bulletin board at 617-369-5622 or call us by telephone—you can reach us at 617-371-2422.

Or fill in this coupon. Whether you order today or not, let us send you full descriptions of our integrated family of Arity products.

arity

We design and distribute high quality, serious application software for the IBM PC, XT, AT and all MS-DOS compatibles.

Please complete this form to place your order and/or request detailed information.

		Quantity	Info only
Arity/Prolog Compiler and Interpreter V4	$795.00	_____	___
Arity/Prolog Interpreter V4	$350.00	_____	___
Arity Standard Prolog	$ 95.00	_____	___
Arity SQL Development Package	$295.00	_____	___
Arity Expert System Development Package	$295.00	_____	___
Arity Screen Design Toolkit	$ 49.95	_____	___
Arity File Interchange Toolkit	$ 49.95	_____	___
TOTAL AMOUNT (MA residents add 5% sales tax) (These prices include shipping to all U.S. cities)		$_____	

NAME_____

SHIPPING ADDRESS_____

CITY/STATE/ZIP_____

TELEPHONE_____

Payment: ☐ Check ☐ PO ☐ AMEX ☐ VISA ☐ MC

Card #_____ Exp. date_____

Signature_____

ARITY CORPORATION • 358 BAKER AVENUE • CONCORD, MA 01742

arity

Circle **no. 121** on reader service card.

Dr. Dobb's Journal of
Software Tools

ARTICLES

Look ma, no retransmissions ▶

TELECOMMUNICATIONS: How to Fix Line Glitches 32
by Joe Marasco
Joe shows how to correct bit errors at the receiving end so that retransmission is not necessary.

Talking to the big boys ▶

TELECOMMUNICATIONS: The CompuServe B Protocol 38
by Levi Thomas and Nick Turner
This public-domain protocol eliminates many of the problems that often plague transmissions between micros and mainframes.

MODULA-2: 68000 Cross Assembler Listings 46
by Brian R. Anderson
The code for the rest of the implementation modules

REVIEWS

Mice and windows are out the door ▶

HUMAN INTERFACE DESIGN: The SwyftCard 42
by Dave Caulkins
Jef Raskin's new user interface for the Apple IIe and IIc

COLUMNS

Powerful sorting without the trimmings ▶

C CHEST: Sort—A General-Purpose Sorting Program 22
by Allen Holub
Faced with his taxes, Allen resorted to a sort utility that can handle files larger than available memory.

16-BIT SOFTWARE TOOLBOX: DUP and FORCDUP Functions, Windows Development Kit, Building Overlays 112
by Ray Duncan
Ray elaborates on some under-documented DOS functions and reports on a seminar for Windows developers. A reader supplies information about the Microsoft linker.

We inaugurate our Structured Programming column ▶

STRUCTURED PROGRAMMING: Overloading Procedures, Exporting Opaque Types, Data Hiding 116
by Namir Clement Shammas
In this issue, Namir addresses Pascal and Modula-2. In the future, other authors will sally Forth.

FORUM

EDITORIAL 6
by Nick Turner

LETTERS 8
by you

CARTOON: Strings Attached 8
by Rand Renfroe

VIEWPOINT: What's Wrong with C 12
by David Carew

DDJ ON LINE: Data Encryption 16

Our editor-in-chief admits his literary lineage ▶

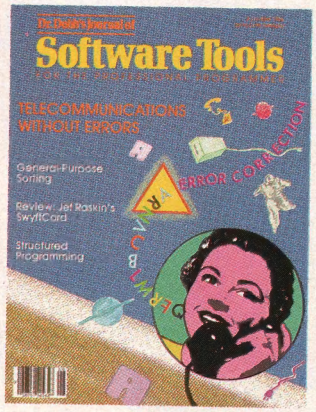
SWAINE'S FLAMES: Carrying the Torch 128
by Michael Swaine

PROGRAMMERS' SERVICES

DR. DOBB'S CATALOG: *DDJ* products—all in one place 73

OF INTEREST: Many new products of interest to programmers 120

ADVERTISER INDEX: Where to find those ads 126



About the Cover
Claudia Steenberg-Majewski pulled this collage out of the blue.

This Issue
This month, we present some relatively painless ways of ensuring error-free telecommunications transmissions. We review Jef Raskin's SwyftCard, a new environment for the Apple IIe and IIc. Especially noteworthy about SwyftCard is that its user interface is philosophically different from that of the Macintosh. Namir Shammas kicks off our Structured Programming column with an invitation to the readers. Michael Swaine, having decided that the back page is his to burn from now on, illuminates his intentions and lights a candle for some of those who have inspired him.

Next Issue
In July, we celebrate Forth. There will be a proposal for a new standard for extended control structures, and we'll shed some light on how to use windows in Forth. Also, in casting about for a novel application, we came up with an implementation in which Forth plumbs the ocean depths. Michael Ham will cover Forth in the July Structured Programming, and even Ray Duncan will get in on the party. For a change of pace, we'll review a number of "turbo" boards for the IBM PC.

YOUR COMPUTER LANGUAGE IS QUIETLY BREEDING REAL BATS IN YOUR BELFRY.

WE'RE OUT TO SAVE ONE MILLION FRUSTRATED PROGRAMMERS

You're on a roll, really pumped, writing the best code you have ever written and then—AAARGHHH!

Freeze dried in your tracks because the language you're using just won't let you achieve what you can conceive.

And you wanted to be a programmer.

So your choices are:

1) write around the problem by creating six pages of emetic code...

2) leave out that incredible idea that really puts your stamp of excellence on this program or...

3) get yourself a world class headache (or a stroke) by dropping into assembler.

Whatever you choose, by now you feel the language is out to get you—because it is.

Sure, no language is perfect, but you have to wonder, "Am I getting all I deserve?"

And, like money, you'll never have enough.

Pretty dismal, huh?

We thought so, too.

So we did something about it.

We call it CLARION.

You'll call it incredible.

With CLARION you can write, compile, run and debug complex applications in a New York afternoon.

Even if you're in Savannah.

It gives you the power and speed to create screens, windows and reports of such richness and clarity you would never attempt them with any other language.

Because YOU would have to write the code.

With CLARION you simply design the screens using our SCREENER utility and then CLARION writes the source code AND compiles it for you.

In seconds.

Likewise, you can use REPORTER to create reports. Remember, only CLARION can recompile and display a screen or report layout for modification.

And with no time wasted.

All the power and facilities you need to write great programs, faster than you ever dreamed of.

Programs that are easy to use.

Programs that are a pleasure to write.

And to you that means true satisfaction.

You've coveted those nifty pop-up help windows some major applications feature. But you can't afford the time and energy it takes to write them into your programs.

That's the way it used to be.

So we fixed that, too.

CLARION HELPER is an interactive utility that let's you design the most effective pop-up help screens that you can imagine. And they're "context sensitive," meaning you can have help for every field in your application.

Unlike the other micro languages, CLARION provides declarations, procedures and functions to process dates, strings, screens, reports, indexed files, DOS files and memory tables.

Imagine making source program changes with the CLARION EDITOR. A single keystroke terminates the EDITOR, loads the COMPILER, compiles the program, loads the PROCESSOR and executes the program. It's that easy!

Our data management capabilities are phenomenal. CLARION files permit any number of composite keys which are updated dynamically.

A file may have as many keys as it needs. Each key may be composed of any fields in any order. And key files are updated whenever the value of the key changes.

Like SCREENER and REPORTER, CLARION's FILER utility also has a piece of the CLARION COMPILER. To create a new file, you name the Source Module. Then you name the Statement Label of a file structure within it.

FILER will also automatically rebuild existing files to match a changed file structure. It creates a new record for every existing record, copying the existing fields and initializing new ones.

Sounds pretty complicated, huh?

Not with CLARION's documentation and on-line help screens. If you are currently competent in Basic, Pascal or "C" you can be writing CLARION applications in a day. In two days you won't believe the eloquence of your CLARION programs.

Okay, now for the best part of all. You can say it in CLARION for $295.00—complete. All you need is an IBM® PC, XT, AT or true compatible, with 320 KB of memory and a hard disk drive.

And we'll allow a full 30 day evaluation period. If you're not satisfied with CLARION, simply return it in its original condition for a full refund.

If you're not ready to take advantage of this no-risk opportunity, ask for our detailed 16 page color brochure. It vividly illustrates the elegance of CLARION. Consider it a preview of programming in the fast lane.

Either way, the call's a freebie.

SAY IT IN CLARION™

Dept. A1ST/1

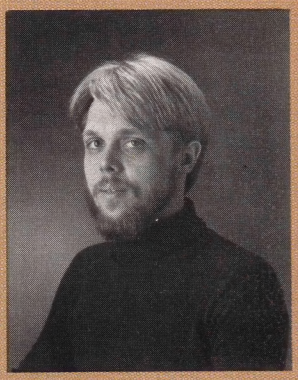
1-800-354-5444

VISA MasterCard AMERICAN EXPRESS

BARRINGTON SYSTEMS, INC. • P.O. BOX 5580 • POMPANO BEACH, FL 33074 • 305/785-4555

IBM® is a registered trademark of International Business Machines Corporation. CLARION™ is a trademark of Barrington Systems, Inc. ©1986 Barrington Systems.

EDITORIAL



Do you know the meaning of "MORF"? Does ":-)" mean anything to you? These are just two of the ASCII shorthand notations sent over the public on-line networks by modem users. A whole new culture seems to be developing in which people who never hear each other's voices meet, get to know each other, and trade even the most intimate of secrets. Anonymity is the rule. You need reveal your true identity only if you want to. The new teleculture is just one facet of the rapidly growing consumer telecommunications industry.

Recent estimates by various sources put the amount of data traffic on the nation's phone networks at 30 percent of total usage. In downtown Manhattan, data traffic is thought to be around 50 percent. Within a few years, those percentages will rise dramatically. Most of the new use is business related, but the consumer market is also expanding rapidly. Recently, CompuServe announced that it now has more than 250,000 subscribers and is gaining between 4,000 and 7,000 new users every month. The Source is adding useful new features (such as special interest groups) and attracting lots of new subscribers. Delphi, People/Link, the WELL (Whole Earth 'Lectronic Link), Dow Jones, and many others are also growing rapidly.

New, exotic features seem to be among the main selling points for the big on-line services. Many boast that they can provide more on-line databases or this encyclopedia or that travel reservation service. Everything from weather forecasting to astrological predictions seems to be available. You can even buy cars, boats, and houses on line.

I can't help but wonder what percentage of the services offered are actually used. What part of the revenues of the big systems actually comes from their "useful" services? I suspect that most of the on-line time is spent in interactive "chat" mode, in which two or more users send lines of text back and forth in real time. I think the next largest amount of time is spent reading messages in SIGs and forums. This is not necessarily a bad sign—in fact, I think it's a sign of the new culture emerging. But the novice may not be getting a very accurate picture of what to expect from some of the current advertisements. People are being led to join on-line services by grand visions that don't necessarily reflect the reality. Why do only a few of the on-line services advertise (and even celebrate) the features that people actually use most?

By the way, "MORF" means "male or female?" and is used as an initial greeting by many chat mode afficionados. The symbol ":-)" is a happy face turned on its side and is appended to a sentence to indicate good feelings or humorous intent, as in "You're such a nerd! :-)" Sometimes the symbol ";-)" is used to show that the sender is winking. Can you guess what "=-0" means?

DDJ is always interested in your article ideas. Right now we're particularly interested in articles for September (algorithms) and October (80286 and 80386). Give me a call at (415) 366-3600 if you've got a nifty idea, or send me a proposal (with an outline, please) at the address in the masthead.

Nick Turner

Dr. Dobb's Journal of
Software Tools

Editorial
Editor-in-Chief *Michael Swaine*
Editor *Nick Turner*
Managing Editor *Vince Leone*
Assistant Editor *Sara Noah Ruddy*
Technical Editor *Allen Holub*
Contributing Editors *Ray Duncan, Allen Holub*
Copy Editor *Rhoda Simmons*
Electronic Editor *Levi Thomas*

Production
Production Manager *Bob Wynne*
Art Director *Shelley Rae Doeden*
Production Assistant *Alida Hinton*
Typesetter *Jean Aring*
Cover Artist *Claudia Steenberg-Majewski*

Circulation
Newsstand Sales Mgr. *Stephanie Barber*
Circulation Director *Maureen Kaminski*
Book Marketing Mgr. *Jane Sharninghouse*
Circulation Assistant *Kathleen Shay*

Administration
Finance Manager *Sandra Dunie*
Business Manager *Betty Trickett*
Accounts Payable Supv. *Mayda Lopez-Quintana*
Accounts Payable Assts. *Denise Giannini, Kathy Robinson*
Billing Coordinator *Laura Di Lazzaro*
Accountant *Marilyn Henry*
Adm. Coordinator *Kobi Morgan*

Advertising Director
Robert Horton (415) 366-3600

Account Managers
Shawn Horst (415) 366-3600
Lisa Boudreau (415) 366-3600
Michele Beaty (317) 875-8093
Michael Wiener (415) 366-3600
Gary George (404) 355-4128
Promotions/Srvcs.Mgr. *Anna Kittleson*
Advertising Secretary *Michelle A. Davié*

M&T Publishing, Inc.
Chairman of the Board *Otmar Weber*
Director *C.F. von Quadt*
President and Publisher *Laird Foshay*

Dr. Dobb's Journal of Software Tools (USPS 3076900 is published monthly by M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600, Second class postage paid at Palo Alto and at additional entry points.

Article Submissions: Send manuscripts and disk (with article and listings) to the Assistant Editor.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0884-5395**

Customer Service: For subscription problems call: outside CA 800-321-3333; within CA 619-485-9623 or 566-6947. For order problems call 415-366-3600.

Subscription Rates: $29.97 per year, U.S. Foreign rates: $56.97, air; $46.97 surface. Foreign subscriptions must be pre-paid in U.S. dollars, drawn on a U.S. Bank. For foreign subscriptions, TELEX: 752-351

Foreign Newsstand Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 6686-1520 TELEX: 620430 (WUI)

Entire contents copyright © 1986 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

ABC Audited Paid Circulation

People's Computer Company
Dr. Dobb's Journal of Software Tools is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit corporation.

The C for Microcomputers

AZTEC

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

Manx Aztec C86

"A compiler that has many strengths ... quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhrystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster, Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/ Link Time
Dhrystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	LN86 Overlay Linker
80186/80286 Support	Librarian
8087/80287 Sensing Lib	Profiler
Extensive UNIX Library	DOS, Screen, & Graphics Lib
Large Memory Model	Intel Object Option
Z (vi) Source Editor -c	CP/M-86 Library -c
ROM Support Package -c	INTEL HEX Utility -c
Library Source Code -c	Mixed memory models -c
MAKE, DIFF, and GREP -c	Source Debugger -c
One year of updates -c	CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System $499
Aztec C86-d Developer's System $299
Aztec C86-p Personal System $199
Aztec C86-a Apprentice System $49

All systems are upgradable by paying the difference in price plus $10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree $395 — **Greenleaf $185**
PHACT $250 — **PC-lint $98**
HALO $250 — **Amber Windows $59**
PRE-C $395 — **Windows for C $195**
WindScreen $149 — **FirsTime $295**
SunScreen $99 — **C Util Lib $185**
PANEL $295 — **Plink-86 $395**

MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

Manx Aztec C68k

"Library handling is very flexible ... documentation is excellent ... the shell a pleasure to work in ... blows away the competition for pure compile speed ... an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C	Creates Clickable Applications
Macro Assembler	Mouse Enhanced SHELL
Overlay Linker	Easy Access to Mac Toolbox
Resource Compiler	UNIX Library Functions
Debuggers	Terminal Emulator (Source)
Librarian	Clear Detailed Documentation
Source Editor	C-Stuff Library
MacRam Disk -c	UniTools (vi,make,diff,grep) -c
Library Source -c	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C86-c systems.

Aztec C68k-c Commercial System $499
Aztec C68d-d Developer's System $299
Aztec C68k-p Personal System $199
C-tree database (source) $399
AMIGA, CP/M-68k, 68k UNIX call

Apple II, Commodore, 65xx, 65C02 ROM

Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3 $399
Aztec C65-d Apple DOS 3.3 $199
Aztec C65-p Apple Personal system $99
Aztec C65-a for learning C $49
Aztec C65-c/128 C64, C128, CP/M $399

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST, Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX ($3000), PDP-11 UNIX ($2000), MS-DOS ($750), CP/M ($750), MACINTOSH ($750), CP/M-68k ($750), XENIX ($750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are $300 to $500 (non VAX) or $1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

CP/M, Radio Shack, 8080/8085/Z80 ROM

Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM) $349
Aztec C II-d (CP/M) $199
C-tree database (source) $399
Aztec C80-c (TRS-80 3 & 4) $299
Aztec C80-d (TRS-80 3 & 4) $199

How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

MANX

To order or for information call:

800-221-0440

UNIX is a registered TM of Bell Laboratories. Lattice TM Lattice Inc., C-tree TM Faircom, Inc., PHACT TM PHACT ASSOC., CI Optimizing C86 TM Computer Innovations, Inc., MACINTOSH, APPLE TM APPLE, INC., Pre-C, PLINK 86, TM PHOENIX, HALO TM Media Cybernetics, Inc., C-terp, PC-lint TM GIMPLE Software, Windscreen, SunScreen TM SunTec, PANEL TM Roundhill Computer Systems Ltd., WINDOWS FOR C TM Creative Solutions, XENIX, MS TM MICROSOFT INC., CP/M TM DRI, AMIGA, C64, C128 TM COMMODORE Int.

LETTERS

Ada

Dear *DDJ*,

As you may be aware, SuperSoft has always been a supporter of *DDJ*. I was particularly interested in the February 1986 issue, which included an article that concerned our Ada compiler. (See "Learning Ada on a Micro.") I was sorry to see, however, that there was no information on how to purchase our Ada in the article. We would like to offer the readers of *DDJ* a 30 percent discount if they mention the magazine when they order Ada by calling (800) 762-6629.

Margie Foote
SuperSoft
P.O. Box 1628
Champaign, IL 61820

STAGE2

Dear *DDJ*,

Years ago, in the late 70s, I used a version of STAGE2, a remarkable macro-converter program, on a DEC PDP-8/E.

More recently, I could see in the CPMUG Library a STAGE2 for the 8080 by Dick Curtiss.

Do you know of the availability of such a macro-converter written for PC-DOS?

Guy Dewarichet
Ave. George Bergmann, 33
B - 1050 Brussels
Belgium

8080 Simulator

Dear *DDJ*,

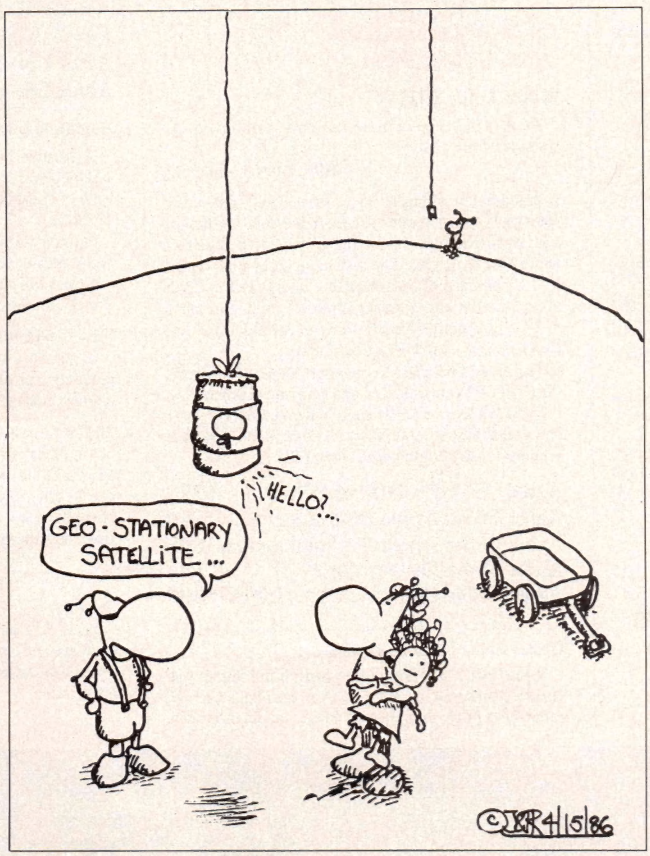
While I was looking through my article "COM: An 8080 Simulator for the MC68000" in *DDJ* (January 1986), I noticed that I had some pretty bad code in the logical instructions. COM originally had all the 8080 registers in memory; with Version 1.2 I moved all the accumulator and flags into 68000 data registers. Unfortunately I didn't take advantage of all the 68000 instructions that I now could. The sequences in Table 1, page 10 (from Version 1.1) could now be written as appears in Table 2, page 10, instead of the way they were published—provided that the high byte of *d0.w* is always assured to be zero. (This is the case with the published code.)

Similar improvements can be made to all *xra*, *ora*, *ana*, *sui*, *ani*, *xri*, and *sub* instructions. *Add* and *adc* instructions don't get shorter because of the *daa* logic, and *sbb* doesn't because of *subx.b* restrictions. The XOR simulations aren't as short as the others are because the 68000 requires the source operand of *eor.b* to be a data register. Along with some short improvements to my *ral*, *rar*, and *daa* instructions (suggested by Edmund Ramm of Germany), changing *dad h* to a shift instruction, and removing an extraneous instruction from *jmp*, I ended up with no perceptable difference! As I had figured before, the real bottlenecks in this program are the opcode dispatcher and the *call*, *jmp*, and *ret* simulations.

The only way I see to really speed this up is with a 68020. As well as having a speed four times faster on 68000 programs, the 68020 has an additional addressing mode of memory indirection that should speed up the opcode dispatcher, and it allows word accesses to odd byte addresses. Table 3, page 10, shows what *call* would be trimmed to.

Perhaps someone with a 68020 machine would care to implement this program and report back the results.

Jim Cathey
ISC Systems Corp.
TAF-C8
Spokane, WA 99220



Inefficient C

Dear *DDJ*,

I'd like to comment on Hal Hardenbergh's Viewpoint column entitled "Inefficient C" in the January 1986 issue of *DDJ*. Although I agree, for the most part, that C isn't as efficient as assembly language, I feel that he overlooked some very important facts:

1. There are many C compilers on the market, particularly for the 8086/8088 processor. The quality of the code produced by these compilers ranges from decent (Manx Aztec C86) to rotten (Lattice C). The size and speed of the code produced by these compilers varies for several reasons, the simplest being that the 8086 has an odd (read: difficult to use) instruction set and architecture (I never liked segmented memory), making optimizer writing a complex task. Other reasons are poor use of registers and high overhead in subroutine calls (especially in programs whose text segments exceed 64K).

On PDP-11-type machines (where C originated), we've found that there is about a 30 percent overhead to C vs. assembler. Most system designers con-

Building an operating system is child's play . . .

. . . with Wendin's Operating System Toolbox™

We know.

We used it to build PCVMS, our $99 version of the versatile VAX/VMS operating system for the IBM PC.

And PCUNIX, the only operating system that puts the powerful features of UNIX on the PC for under $100.

But we didn't create this powerful software construction set just for us. The toolbox is for any programmer who wants to build his or her own multitasking, multiuser operating systems.

Systems compatible with the PC-DOS file system and with most MS-DOS programs.

In nine basic modules, toolbox provides everything you need (including source and object code) to build a personal operating system that fits you and your work habits — instead of the other way around.

And to help you get started, we've written a step-by-step instruction manual that shows you how to write a shell and link it with the toolbox kernel.

The only thing we don't provide is a creative imagination. If you've got that, the rest is child's play.

Operating System Toolbox From Wendin. Only $99.

Circle **no. 112** on reader service card.

WENDIN®
BOX 266
CHENEY, WA 99004

© Copyright 1986 Wendin, Inc.

The people who make quality software tools affordable.

ORDER HOTLINE

(509) 235-8088

(MON.-FRI., 8-5 PACIFIC TIME)

VISA MasterCard

Ask about our other products for the IBM PC and true compatibles.

PCUNIX™
True multitasking, multiuser operating system similar to AT&T's popular UNIX™ operating system.

PCVMS™
Multitasking, multiuser version of DEC's powerful VAX/VMS operating system. Runs most MS-DOS programs.

XTC®
The ultimate programmer's editor. Multitasking macro language plus multiple linkable windows and buffers.

All products priced at $99 with source code included.

DEALER INQUIRIES WELCOME

Foreign orders inquire about shipping.
Domestic orders add $5.00/1st item, $1.00 each additional item for shipping handling, and insurance.
Washington residents add 7.8% sales tax.

MS is a trademark of Microsoft; PC-DOS is a trademark of IBM; UNIX is a trademark of AT&T; VAX/VMS is a registered trademark of Digital Equipment Corporation

Wendin and XTC are registered trademarks of Wendin, Inc.
PCUNIX, PCVMS, Operating System Toolbox, and Personal Operating System are trademarks of Wendin, Inc.

(continued from page 8)

sider this quite good compared to other high-level languages (even, dare I say it in the same breath, FORTRAN).

On other types of architectures, C is better or worse depending on how well the instruction set matches the operators in C, and on how much time and effort is put into the compiler design and the optimizer.

2. The paragraph that talks about the market voting with its wallet and not caring about "how hard it was to produce a program or how long it took," etc., is true in essence. Users are interested in two basic factors when purchasing software: cost and functionality. Speed is an important factor in functionality. Mr. Hardenbergh has obviously not gone to market with many products, however; otherwise he'd know that the "market window" makes or breaks a product.

A program that works, however slowly, is better than one that is still being written. Being first can be much more important than being fastest! Prototyping in C and rewriting parts in assembly language is an accepted method of software design; it also allows a product (alpha or beta version) to be placed in the market ASAP. If coding in C can reduce the development time for a product, then this may also bring down the cost so that even if the product is slower than its assembly sibling, it will be less expensive. (Somehow, this doesn't seem to happen, though. I wonder if corporate greed enters into play here?)

3. An example of an application in which C has made a firm stand is in the area of operating systems. Consider that the Unix system is about 90 percent C and 10 percent assembly (interesting that this particular ratio pops up, isn't it?); it comprises about 100K of instructions and about another 100K of data (give or take a little depending on the number of device drivers installed and the amount of memory devoted to buffer caches). Now consider that mainframe operating systems written in assembly are much larger (MVS is around 130K without TSO, which is necessary if you want to have an interactive system.)

Whether Unix is more or less functional than other operating systems is a long-standing dispute; however, in looking at Amdahl's UTS system (a Unix System V implementation for IBM mainframes), I've seen a system that can support more users than can MVS/TSO or VM/CMS on the same processor. Not only is UTS faster, but it also has a feature that no IBM mainframe has: full-duplex asynchronous communications (which we're so used to that we forget how annoying half-duplex is).

I guess that the point here is that even though Unix is coded mostly in C, it has enough functionality to make a dent in a marketplace dominated by products coded in assembly language; enough functionality to force companies to offer it as an option even though it competes with their own operating systems.

Anyway, there is no question that assembly produces faster code than C does in practically every application; the questions are whether the overhead that goes along with C is worth the reduction in development time and overall product cost and whether having the time to add greater functionality to the product is desirable.

Patrick Wood
Pipeline Associates Inc.
49 Manito Ave.
Lake Hiawatha, NJ 07034

Correction

Listing Five of Brian R. Anderson's article, "A 68000 Cross Assembler—Part 1," (April 1986) was incomplete. The complete listing is shown in Table 4, below.

DDJ

```
and b       move.b regb(regs),d0              ; A0 Ana B
            and.b rega(regs),d0
            move.b d0,rega(regs)
            and.w regconff,d0
            move.b 16(flagptr,d0.w),regf(regs)
            jmp (return)
```

Table 1: *The original AND*

```
and.b       and.b regb(regs),rega             ; A0 Ana B
            move.b 16(flagptr,rega.w),regf
            jmp (return)
```

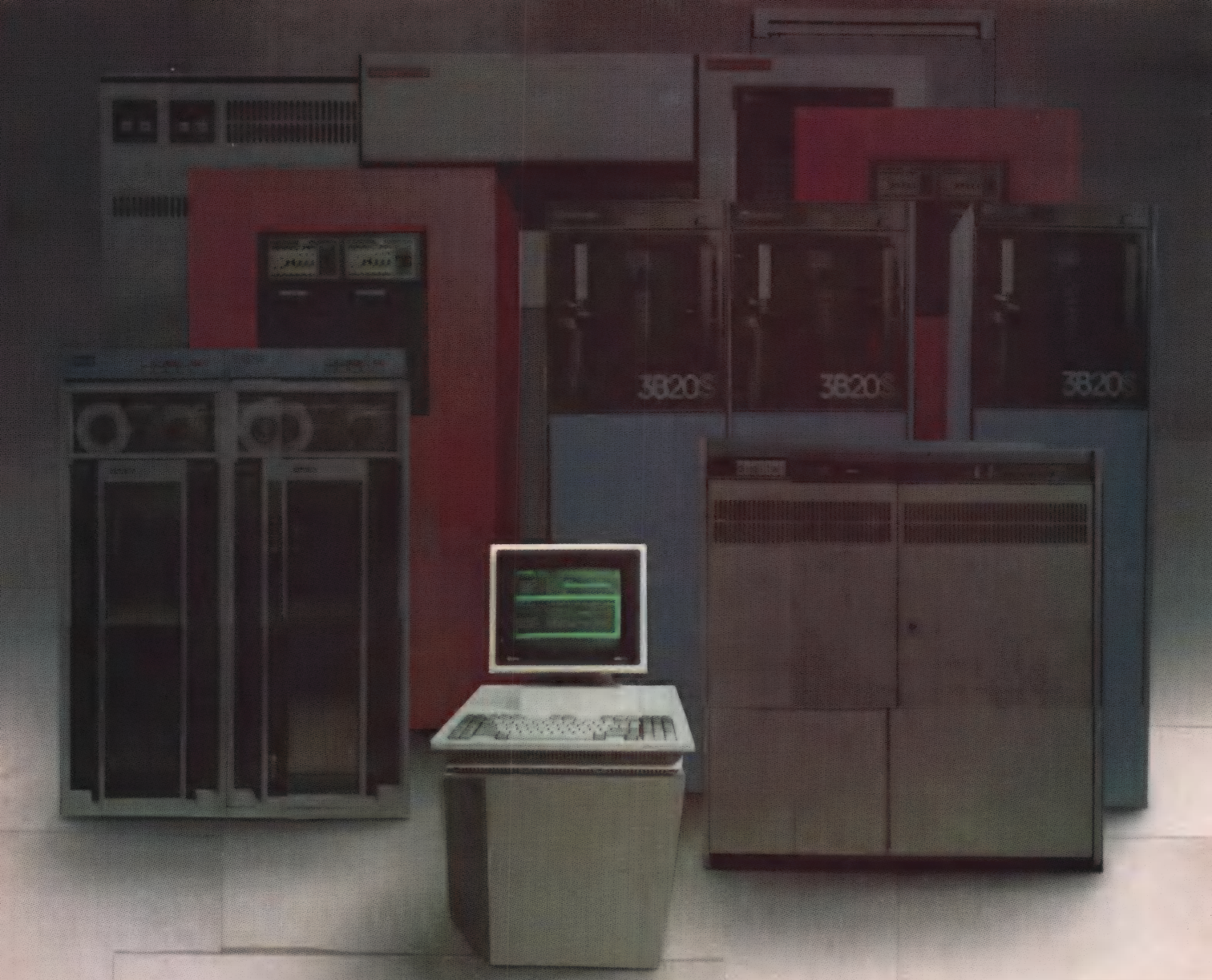
Table 2: *The improved AND*

```
call        move.w (pseudopc)+,d0
            rol.w #8,d0             ; Byte reversal, but
            move.1 pseudopc,d1
            sub.1 targbase, d1
            rol.w #8,d1             ; barrel shifter is quick!
            move.w d1,-(sp)
            lea.l 0(targbase,d0.1),pseudopc
            jmp (return)
```

Table 3: *CAll using 68020 instructions*

```
DEFINITION MODULE CodeGenerator;
(* Uses information supplied by Parser, OperationCodes, *)
(* and SyntaxAnalyzer to produce the object code.       *)

   FROM Parser IMPORT
      TOKEN, OPERAND;

   FROM LongNumbers IMPORT
      LONG;


   EXPORT QUALIFIED
      LZero, AddrCnt, Pass2, BuildSymTable, AdvAddrCnt, GetObjectCode;


   VAR
      LZero, AddrCnt : LONG;
      Pass2 : BOOLEAN;


   PROCEDURE BuildSymTable (VAR AddrCnt : LONG;
                            Label, OpCode : TOKEN; SrcOp, Destp : OPERAND);
   (* Builds symbol table from symbolic information of Source File *)

   PROCEDURE AdvAddrCnt (VAR AddrCnt : LONG);
   (* Advances the address counter based on the length of the instruction *)

   PROCEDURE GetObjectCode (Label, OpCode : TOKEN;
                            SrcOp, DestOp : OPERAND;
                            VAR AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
                            VAR   nA,      nO,      nS,      nD,    : CARDINAL);
   (* Determines the object code for the operation as well as the operands *)
   (* Returns each (up to 3 fields), along with their length              *)


END CodeGenerator.
```

Table 4



UNIX™ SYSTEM POWER FOR PEOPLE WITH BIGGER THINGS IN MIND.

You know what **UNIX**™ System V can do.

But now you don't need a mini to do it. The AT&T **UNIX** PC puts room-size computing power right on a desktop.

Its Motorola 68010 chip, 10 MHz clock speed and up to 4MB RAM—with virtual memory support and internal hard disk options from 10 to 67MB—give you 75% of the power of a VAX* 11/780.

For only 7% of the cost.

Development tools? The AT&T **UNIX** PC puts you in a **UNIX** System V environment complete with system utilities, the shell, C compiler and 68010 assembler. As for languages, you get the full range: C, Cobol, Fortran, Pascal, BASIC and the LPI** high-performance suite. Not to mention C-ISAM†, INFORMIX† and sort/merge for database development.

All, with the convenience of built-in text editors, debuggers and graphics tools, including the GSS Virtual Device Interface.

Up- and downloading your work from minis or mainframes is easy. Thanks to the standard internal 300/1200 bps modem, RS-232 port, VT 100* terminal emulation software and optional 3270 terminal emulation. You also get two jacks for phone lines and built-in communications software.

All of which make the AT&T **UNIX** PC ideal for ongoing voice/data communications and remote access to shared corporate databases.

ONE OF THE COMPUTERS WITH THE FUTURE BUILT IN.

Even with all its available power and storage options, the AT&T **UNIX** PC still has room to grow. With three expansion slots and the ability to connect up to seven serial devices.

Because when you have big ideas, accommodating them shouldn't be a big deal.

To find out about the AT&T **UNIX** PC and our **SPECIAL LIMITED TIME OFFER** call your AT&T Account Executive, authorized AT&T supplier or 1 800 247-1212.

AT&T

The right choice.

*VAX and VT100 are trademarks of Digital Equipment Corporation. **LPI is a trademark of Language Processors, Inc.
†C-ISAM and INFORMIX are trademarks of Relational Database Systems, Inc. © 1986 AT&T Information Systems.

VIEWPOINT

What's Wrong with C

by David Carew

David Carew is a systems analyst developing banking applications at Inc., 2864 South Circle Dr., Ste. 200, Cheyenne Centre, Colorado Springs, CO 80906.

Conventional wisdom is something that ought to be questioned periodically. The selection of C as the language of choice for microcomputer system software development has arguably attained the status of conventional wisdom. After all, what is good enough for Bill Gates and his crew ought to be good enough for us, right?

I realize that criticizing a programmer's favorite language is likely to provoke a defensive reaction more visceral than rational, and I expect some controversy.

My first criticism is that the code produced by most people using C as a tool is (hang on to your hat!) bulky and slow. Furthermore, it is not just bulky and slow compared with assembly language; it is inefficient compared with the output of an average production-quality optimizing compiler.

Conventional wisdom Lsays that the reward for working in an inherently low-level language such as C is efficiency. This is not necessarily so. In C, as in assembly language, optimization is the responsibility of the individual programmer. The whole C philosophy would have you believe this is the correct emphasis. Unfortunately hand optimization, like documentation, never gets done (unless of course, the product is about to become obsolete).

The brutal fact is that an average optimizing compiler will outdo the hand-coded assembler implementation of 80 percent of the programming population; the other 20 percent would take from three to ten times longer to get a better implementation up and running. A second brutal fact is that C, with its low-level philosophy and direct implementation of pointers and machine-level constructs, simply doesn't allow the use of standard compiler optimization techniques. Object code generated by a C compiler almost never beats hand-coded assembly.

If better quality compilers were demanded, better quality would be delivered. The fact that good quality optimizing compilers seem scarce in the microcomputer market should not be an excuse for sticking with C.

My second criticism of C is also directed at something usually regarded as a strength, or at least as an opportunity for the proverbial "experienced programmer": C's operator set is too rich. Taken with the operator characters in C's omnipresent preprocessor, all those special operators (++, &, *, --, and so forth) and their accompanying precedence rules form a little "language within a language." The programmer is rewarded for knowing the nuances and tricks of this "language"—rewarded with much more efficient code.

Thus guided by the invisible hand of the compiler, the programmer inevitably tends: (a) to write less understandable, less supportable code; and (b) to become distracted from the task of contriving an optimal solution to the problem at hand. I am often struck by the impression that a given C program is an elegant example of C and its operators but misses the point as a solution. On the one hand, here is a concordance generator that builds a binary tree instead of using a faster, shorter, and more robust sort, but its use of pointer operators in building the tree is expertly done. On the other hand, there is a *grep*-style search utility in which expert use of C's nuances is made in the service of a hard-wired, "look-ahead" style parser that is admittedly slower (and probably bulkier) than a good table-driven parser.

From the standpoint of the software designer, a good compiler might allow the programmer to say either

```
b = ++i
```

or

```
i = i+1
b = i
```

as long as the object code generated is the same. When one construct generates radically better code (and when there are dozens or hundreds of such tricks and trade-offs), it is natural for the programmer to expend effort optimizing his or her use of the progamming notation as well as devising an efficient solution to the problem at hand. Because the compiler is well-defined, consistent, and approachable and the application problem is likely to be ill-defined, inconsistent, and messy, it is very easy for the emphasis to become misplaced.

Make no mistake about it, better algorithms and data structures for solution of the application problem are far more important than is ideal use of a complex programming notation. To see this for yourself, benchmark a quick-and-dirty, compiled BASIC quick sort against the tightest, best-coded, C language selection sort you can devise or find.

C is tending to create a new computer elite, a barrier to those who haven't the time or inclination to master its complexity (and, devotees would say, its attendant "power"). Because the investment in learning C is so high, there is a strong psychological addiction factor. C wizards like being C wizards, and the sociology of this understandable bias may be dangerously close to creating an unnecessary, artificial barrier to further progress in microcomputer software technology.

The measure of a compiler-based programming language is in the quality of its output object code and, perhaps even more important, in the productivity of average (nongenius, nonwizard) programmers, who produce and maintain the vast bulk of all source code in any language. By these measures, C is a surprisingly poor language, given its unquestioned acceptance. It may be the best choice we have at the moment—though even that contention may be open to debate.

DDJ

Turbo Pascal and the Turbo Pascal family give you a perfectly integrated programming environment and unbeatable speed, power, and price

Turbo Pascal® is *faster* than any other Pascal compiler, and at only $69.95, a distinctly better deal. But it offers much more than speed, power, and price.

There's also the complete Pascal family of products that's grown from 1 to 9 products in just 3 years.

- Turbo Pascal 3.0
- Turbo Pascal with the 8087 support
- Turbo Pascal with Binary Coded Decimal, (BCD)
- Turbo Pascal with 8087 *and* BCD
- Turbo Database Toolbox™
- Turbo Graphix Toolbox™
- Turbo Tutor®
- Turbo Editor Toolbox
- Turbo GameWorks

Turbo Pascal is backed by a complete range of "toolboxes" that give you most of the programming tools you'll ever need.

The Turbo Pascal family is never static, but is continuously expanding, with new products like Turbo Editor Toolbox™ and Turbo Gameworks.™

The secret of software success is not merely low price, but top quality, allied with complete documentation, like our 400-page reference manual.

All of which are some of the reasons why Turbo Pascal is clearly the leader, and the recipient of awards like PC Week's **"Product of the Year"** and PC Magazine's **"Award for Technical Excellence."** And some of the reasons why Turbo Pascal has now become a *de facto* worldwide standard with more than half a million users.

Turbo Pascal has grown from a single product 3 years ago to a family of 9 today.

Success breeds success, so the Turbo Pascal family has flourished. Your choices now include:

☐ **Turbo Pascal 3.0** combines the fastest Pascal compiler with an integrated development environment.

☐ **Turbo Pascal with 8087** math co-processor support for heavy duty number-crunching, and/or **Binary Coded Decimals** to eliminate rounding-off errors for business applications.

☐ **Turbo Database Toolbox** is a perfect complement to Turbo Pascal. It includes a complete library of Pascal procedures that allows you to search and sort data, and build powerful database applications.

☐ **Turbo Graphix Toolbox** includes a library of graphics routines for Turbo Pascal programs. Lets even beginning programmers create high-resolution graphics with an IBM,® Hercules,™ or compatible graphics adapter. Does complex business graphics, easy windowing, and stores screen images to memory.

☐ **Turbo Tutor** teaches you step by step how to use Turbo Pascal, with commented source code for all program examples on diskette.

Save $109.70 when you choose the Turbo Jumbo Pack. 6 different Turbo Pascal products for only $245.00!

For only $245.00, you get Turbo Pascal 3.0 *and* Turbo Editor Toolbox *and* Turbo Tutor *and* Turbo Graphix Toolbox *and* Turbo GameWorks *and* Turbo Database Toolbox!

All 6 for only $245.00, which saves you $109.70. This limited offer is good through September 1, 1986, so act now.

NEW! *Amazing value! Turbo Editor Toolbox includes MicroStar,™ a full-blown editor that also does windows!*

Turbo Editor Toolbox not only gives you ready-to-compile source code and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs, but also includes MicroStar, a complete editor with full windowing capabilities. (You could pay $100.00 or more for a program like MicroStar, but you get it free as part of our Turbo Editor Toolbox.) You can also use Turbo Editor (which of course integrates with Turbo Lightning™) to build your own word processor!

NEW! *Turbo GameWorks gives you the games you can write, rewrite, bend and amend!* **Turbo GameWorks** reveals the secrets of game design and the strategies. You're given source code, a 200-page manual, and the insight needed to write and customize your own irresistible games.

Turbo GameWorks also includes ready-to-play Chess, Bridge, and Go-Moku—an ancient Japanese game that can divert you from reality for hours on end.

"Language deal of the century . . . Turbo Pascal
Jeff Duntemann, PC Magazine

Turbo Pascal has got to be the best value in languages on the market today
Jerry Pournelle, BYTE Magazine

This compiler, produced by Borland International, is one of the best programming tools presently available for the PC
Michael Covington, PC Tech Journal

YES! I want the best

To order by phone, or for a dealer nearest you,
call (800) 255-8008
in CA call (800) 742-1133

Copies	Product	Price	Totals
___	Turbo Pascal 3.0	$69.95	$ ___
___	Turbo Pascal w/8087††	$109.90	$ ___
___	Turbo Pascal w/BCD††	$109.90	$ ___
___	Turbo Pascal w/8087, BCD††	$124.95	$ ___
___	Turbo Database Toolbox	$54.95	$ ___
___	Turbo Graphix Toolbox†	$54.95	$ ___
___	Turbo Tutor	$34.95	$ ___
___	Turbo Editor Toolbox†	$69.95	$ ___
___	Turbo GameWorks†	$69.95	$ ___
___	Turbo Jumbo Pack†	*$245.00	$ ___
	Outside USA add $10 per copy CA and MA res. add sales tax		$ ___
	Amount enclosed		$ ___

Prices include shipping to all US cities.

Carefully describe your computer system:

Mine is: ___ *8-bit* ___ *16-bit*

I use: ___ *PC-DOS* ___ *MS-DOS* ___ *CP/M-80* ___ *CP/M-86*

My computer's name and model is: _____

The disk size I use is: ☐ *3½"* ☐ *5¼"* ☐ *8"*

Payment: *VISA* *MC* *Bank Draft* *Check*

Credit card expiration date ___/___

Card # _____

NOT COPY PROTECTED TF11

****60-DAY MONEY-BACK GUARANTEE**

Name: _____

Shipping Address: _____

_____

City: _____

State: _____ *Zip:* _____

Telephone: _____

CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by credit card or International Postal Money Order.

**Limited Time Offer until September 1, 1986.*

***YES, if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.*

Minimum System Requirements:
Turbo GameWorks, Turbo Graphix Toolbox, & Turbo Editor Toolbox—192K. All other products, 128K.
†IBM PC, PCjr, AT, XT, and true compatibles.
††16-bit only.

BORLAND INTERNATIONAL
Vive la différence

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Borland products include Turbo Pascal; Turbo Prolog; Turbo Database Toolbox; Turbo Lightning; Turbo Graphix Toolbox; Turbo Tutor; Turbo GameWorks; Turbo Editor Toolbox; Word Wizard; Reflex, The Analyst; SideKick; Sidekick, The Macintosh Office Manager; Traveling SideKick; and SuperKey—all of which are trademarks or registered trademarks of Borland International, Inc. or Borland/Analytics, Inc.

Turbo Pascal and Turbo Tutor are registered trademarks, and Turbo GameWorks, Turbo Editor Toolbox, Turbo Database Toolbox, Turbo Graphix Toolbox, Turbo Lightning, and MicroStar are trademarks of Borland International. IBM is a registered trademark of International Business Machines Corp. Hercules is a trademark of Hercules Computer Tech.
Copyright 1986 Borland International. BI-1036C

Ada just moved into a smaller place.

We have some very good news for you.

You can now get a validated, full Ada® compiler for the IBM® PC AT. From the people who designed the Ada language. For just $3,000.

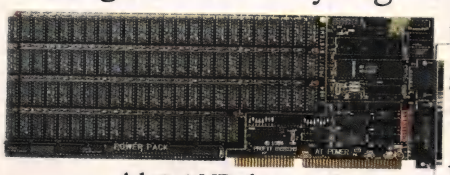
Which means you and your company can now program in Ada without tying up a big, expensive computer.

And you should program in Ada. And not just because the DoD says so.

The DoD mandates Ada for their software principally because Ada is considerably easier and less expensive to maintain.

Does more reliable and easier to maintain code sound attractive to you? If not, just look at how your programmers are spending 80% of their time.

People who know Ada are calling it "the only logical language for the eighties and nineties." The point is, they're not thinking of Ada as "the DoD language." It's simply because Ada supports good, solid software engineering practice.

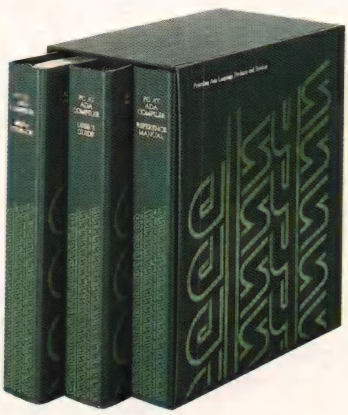


Ada & 4-MB of memory*
for the price of 4-MB of memory.

And now you can try full Ada programming for less than the cost of a two week training program.

The Alsys™ Ada compiler for the PC AT is not only validated, it's actually written in Ada. And produces code so efficient it executes faster than C or Pascal on tested benchmarks.

And if that's not enough, the Alsys PC AT Ada compiler runs in protected mode. So you can use the full amount of memory available to the PC AT. This means you can run a program using up to 16 megabytes of memory for code and data without worrying about DOS, overlays and all that stuff.



Lastly, this compiler comes with a 4-megabyte memory upgrade board. That, by itself, is worth the price of admission.

At this point, we suspect you might be tempted to pull out a credit card and give us a call.

Or at least fill out the coupon.

Write: Alsys, Inc., 1432 Main Street, Waltham, MA 02154, U.S.A., Telephone: (617) 890-0030, Telex: 948536.

In France: Alsys, S.A., 29, Avenue de Versailles, 78170 La Celle St. Cloud, France, Telephone: 33(1)3918 12 44, Telex: 697569.

In England: Alsys, Ltd., Partridge Hse, Newtown Road, Henley-on-Thames, Oxon RG9 1EN, England, Telephone: 44 (491) 579090, Telex: 846508.

alsys

1432 Main Street, Waltham, MA 02154

☐ Send me more information about the Alsys PC AT Ada compiler.
☐ Send me more information about Ada.

Name

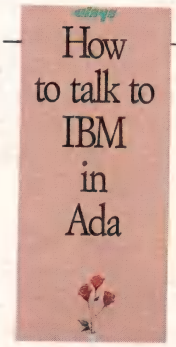
Title

Company

Address

City State Zip

Telephone



Knowledge is good.
Especially when it's free.

Ada© is a registered trademark of the U.S. Government (Ada Joint Program Office). IBM® and PC AT are registered trademarks of International Business Machines Corporation. Alsys™ is a trademark, service mark, and trade name of Alsys S.A. *Memory board manufactured by Profit Systems, Inc.

Circle **no. 273** on reader service card.

DDJ ON LINE

The following was written and uploaded to Data Library 0 by Forum member John A. Thomas. The online discussion ignited by this article and tended by John, which is in DL0 as well (Type KEYWORDS:ENCRYPT THREAD*), is generating a good deal of heat and light. Feel free to add your comments to the message board there.*

Survey of Data Encryption

This article is a survey of data encryption. It is intended to provoke discussion among the members of this forum and perhaps lead to a creative exchange of ideas. Although the basics of the subject seem to be known to few programmers, it embraces many interesting and challenging programming problems, ranging from the optimization of machine code for maximum throughput to the integration of encryption routines into editors, communications packages, and perhaps products not yet invented. Governments have dominated this technology until the last few years, but now the need for privacy and secrecy in the affairs of a computer-using public has made it essential that programmers understand and apply the fundamentals of data encryption.

by John A. Thomas
CIS 75236,3536

Some Cryptographic Basics

A few definitions are appropriate first. We use the term *encryption* to refer to the general process of making plain information secret and making secret information plain. To *encipher* a file is to transform the information in the file so that it is no longer directly intelligible. The file is then said to be in *ciphertext*. To *decipher* a file is to transform it so that it is directly intelligible—that is, to recover the *plaintext*.

The two general devices of encryption are *ciphers* and *codes*. A cipher works on the individual letters of an alphabet, whereas a code operates on some higher semantic level, such as whole words or phrases. Cipher systems may work by transposition (shuffling the characters in a message into some new order), by substitution (exchanging each character in the message for a different character according to some rule), or by a combination of both. In modern usage, transposition is often called permutation. A cipher that employs both transposition and substitution is called a *product* cipher. In general, product ciphers are stronger than those using transposition or substitution alone. Shannon[1] refers to substitution as "confusion" because the output is a nonlinear function of the input, thus creating confusion as to the set of input characters. He referred to transposition as "diffusion" because it spreads the dependence of the output from a small number of input positions to a larger number.

Every encryption system has two essential parts: an algorithm for enciphering and deciphering and a *key*, which consists of information to be combined with the plaintext according to the dictates of the algorithm. In any modern encryption system, the algorithm is assumed to be known to an opponent, and the security of the system rests entirely in the secrecy of the key.

Our goal is to translate the language of the plaintext to a new "language" that cannot convey meaning without the additional information in the key. Those familiar with the concept of entropy in physics may be surprised to learn that it is also useful in information theory and cryptography. Entropy is a measure of the amount of disorder in a physical system or the relative absence of information in a communication system. A natural language such as English has a low entropy because of its redundancies and statistical regularities. Even if many of the characters in a sentence are missing or garbled, we can usually make a good guess as to its meaning. Conversely, we want the language of our ciphertext to have as high an entropy as possible; ideally, it should be utterly random. Our guiding principle is that we must increase the uncertainty of the cryptanalyst as much as possible. His uncertainty should be so great that he cannot make any meaningful statement about the plaintext after examining the ciphertext; also, he must be just as uncertain about the key, even if he has the plaintext itself and the corresponding ciphertext. (In practice, it is impossible to keep all plaintext out of his hands.)

A prime consideration in the security of an encryption system is the length of the key. If a short key (that is, short compared with the length of the plaintext) is used, then the statistical properties of the language will begin to "show through" in the ciphertext as the key is used over and over, and a cryptanalyst will be able to derive the key if he has enough ciphertext to work with. On the other hand, we want a relatively short key so that it can be stored easily or even be remembered by a human. The government or a large corporation may have the means to generate and store long binary keys, but we cannot assume that the personal computer user will be able to do so.

The other important fact about the keys is that there must be very many of them. If our system allows only 10,000 different keys, for example, it is not secure because our opponent could try every possible key in a reasonable amount of time. This introduces the concept of the "work factor" required to break an encryption system. We may not have a system unbreakable in principle, but if we can make the work factor for breaking so high it is not practical for our opponent to do so, then it is irrelevant that the system may be less strong than the ideal. What constitutes an adequate work factor depends essentially on the number of uncertainties the cryptanalyst must resolve before he can derive plaintext or a key. In these days of constantly improving computers, that number should probably exceed 2^{128}. It is easy to quantify the work factor if we are talking about exhaustive key trial, but few modern ciphers are likely to be broken by key trial because it is too easy to make the key space very large. Most likely they will be broken be-

QUIT DOING GRUNT WORK.

Let Greenleaf do it for you and set you free.

C Program developers, stop slaving!

Greenleaf libraries have the functions you need — already perfected and in use by winning program developers in major corporations such as IBM, EDS and GM.

Between our Greenleaf Functions and Greenleaf Comm Library, we have over 340 functions on the shelf. Each one can save you time and effort. Money, too.

Many C programmers have told us that, even if they only use one or two functions, our products easily pay for themselves:

The Greenleaf Functions

The most complete and mature C language function library for the IBM PC, XT, AT and close compatibles. Our version 3.0 includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, new disk status and Ctrl-Break control functions plus many more!

The Greenleaf Comm Library

Our 2.0 version is the hottest communications facility of its kind. Over 120 all new functions — ring buffered, interrupt-driven asynchronous communications.

Call Toll Free

1-800-523-9830

In Texas and Alaska, call

214-446-8641

GREENLEAF Software

Greenleaf Software, Inc.
1411 LeMay Drive Suite 101
Carrollton, TX 75007

If you need more than 2 ports, only Greenleaf gives you the total solution — boards, software, and complete instructions that enable you to build a 16-port communication system.

And no matter how many ports you have, it's virtually impossible to lose information with multiple file transfers. XMODEM, XON/XOFF and Hayes modem controls are featured.

We support all popular C compilers for MS DOS: Lattice, Microsoft, Computer Innovations, Wizard, Aztec, DeSmet and Mark Williams.

Order today!

Order a Greenleaf C library now. See your dealer or call 1-800-523-9830. Specify compiler when ordering. Add $8 for UPS second day air, or $5 for ground. Texas residents, add sales tax. Mastercard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf Comm Library v2.0	$185
Greenleaf Functions v3.0	$185
Digiboard Comm/4-II	$315
Digiboard Comm/8-II	$515

We also sell compilers, books and combination packages.

Circle **no. 97** on reader service card.

cause of internal periodicities and subtle dependency of output on input, which give the cryptanalyst enough information to reduce his uncertainty by orders of magnitude.

A corollary to work factor is the rule that a system need only be strong enough to protect the information for however long it has value. If a system can be broken in a week, but not sooner, then it may be good enough if the information has no value to an opponent after a week.

Cryptanalysis

Cryptanalysis is the science of deriving plaintext without the key information. Anyone intending to design an encryption system must be acquainted to some degree with cryptanalytic methods. The methods of attack may range from sophisticated statistical analysis of ciphertext to breaking into the opponent's office and stealing his keys ("practical cryptanalysis"). There are no rules of fair play. The cryptanalyist is free to use his puzzle-solving ingenuity to the utmost, even to the point of applying the knowledge that your dog's name is Pascal and that you might be lazy enough to use that as your key for the day.

The cryptanalyst may have only ciphertext to work with, he may have both ciphertext and the corresponding plaintext, or he may be able to obtain the encipherment of chosen plaintext. Some cryptographic systems are fairly strong if the analyst is limited to ciphertext but fail completely if he has corresponding plaintext. Your system should be strong enough to resist attack even if your opponent has both plaintext and ciphertext.

Computer power can greatly aid cryptanalysis, but many systems that appear strong can be broken with pencil-and-paper methods. The Vigenere family of polyalphabetic ciphers, for example, was generally believed to be unbreakable up until the late nineteenth century. A polyalphabetic cipher is a substitution cipher in which a different alphabet is used for each character of plaintext. In these systems, the key determines the order of the substitution alphabets, and the cycle repeats with a period equal to the length of the key. This periodicity is a fatal weakness because fairly often a repeated letter or word of plaintext will be enciphered with the same key letters, giving identical blocks of ciphertext. This exposes the length of the key. Once we have the length of the key, we use the known letter frequencies of the language to gradually build and test hypotheses about the key. Vigenere ciphers can be implemented easily on computers, but they are worthless today. Designers without knowledge of cryptanalysis, however, might be just as ignorant of this fact as their colleagues of the last century. Please see the references at the end of this article for information on cryptanalytic techniques.

A Survey of Cryptographic Systems

I'll now review some representative encryption schemes, starting with traditional ones and proceeding to the systems that are only feasible when implemented on computers.

The infinite-key cipher, also known as the "one time pad," is simple in concept. First, we generate a key that is random and at least the same length as our message. Then, for each character of plaintext, we add the corresponding character of the key to give the ciphertext. By addition, we mean some reversible operation; the usual choice is the exclusive-OR. A little reflection will show that, given a random key at least the size of the plaintext (that is, "infinite" with respect to the plaintext because it is never repeated), the resulting cipher is unbreakable, even in principle. This scheme is in use today for the most secret government communications, but it presents a serious practical problem with its requirement for a long random key for each message and the need to somehow send the lengthy key to the recipient. Thus the ideal infinite-key system is not practical for large volumes of message traffic. It is certainly not practical for file encryption on computers because where would the key be stored? Be wary of schemes that use software random-number generators to supply the infinite key. Typical random-number algorithms use the preceding random number to generate the succeeding number and can thus be solved if only one number in the sequence is found.

Some ciphers have been built to approximate the infinite-key system by expanding a short key. The Vernam system for telegraph transmission used long paper tapes containing random binary digits (Baudot code, actually) that were exclusively-ORed with the message digits. To achieve a long key stream, Vernam and others used two or more key tapes of relatively prime lengths, giving a composite key equal to their product. The system is still not ideal because eventually the key stream will repeat, allowing the analyst to derive the length and composition of the keys given enough ciphertext. There are other ways to approach the infinite-key ideal, some of which are suggested in my article (with Joan Thersites) in the August 1984 issue of *DDJ*. (See sidebar on page 20.)

Rotor systems take their name from the electromechanical devices of World War II, the best known being perhaps the German ENIGMA. The rotors are wheels with characters inscribed on their edges and with electrical contacts corresponding to the letters on both sides. A plaintext letter enters on one side of the rotor and is mapped to a different letter on the other side before passing to the next rotor and so on. All the rotors (and there may be few or many) are then stepped so that the next substitution is different. The key is the arrangement and initial setting of the rotor disks. These devices are easy to implement in software and are fairly strong. They can be broken, however; the British solution of the ENIGMA is an interesting story outside the scope of this article. If you implement a rotor system, consider having it operate on bits or nybbles instead of bytes, consider adding permutation stages, and consider how you are going to generate the rotor tables because you must assume these will become known to an opponent.

In 1977 the National Bureau of Standards promulgated the Data Encryption

PERFORMANCE PACKAGE

Blaise Computing Inc. introduces the PERFORMANCE PACKAGE™ for Turbo Pascal programmers.

TURBO PASCAL

NEW!

Turbo ASYNCH™

With Turbo ASYNCH, you can be in constant touch with the world without ever leaving the console. Rapid transit at its best. Turbo ASYNCH is designed to let you incorporate asynchronous communication capabilities into your Turbo Pascal application programs, and it will drive any asynchronous device via the RS232 ports, like printers, plotters, modems or even other computers. Turbo ASYNCH is fast, accurate and lives up to its specs. Features include...

◆ Initialization of the COM ports allowing you to set all transmission options. ◆ Interrupt processing. ◆ Data transfer between circular queues and communications ports. ◆ Simultaneous buffered input and output to both COM ports. ◆ Transmission speeds up to 9600 Baud. ◆ Input and output queues as large as you wish. ◆ XON/XOFF protocol.

The underlying functions of Turbo ASYNCH are carefully crafted in assembler for efficiency, and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The interface to the assembler routines is written in Turbo Pascal.

The Turbo Pascal PERFORMANCE PACKAGE™ is for the serious Turbo Pascal programmer who wants quality tools to develop applications. Every system comes with a comprehensive User Reference Manual, all source code and useful sample programs. They require an IBM PC or compatible, utilizing MS-DOS version 2.0 or later. There are no royalties for incorporating PERFORMANCE PACKAGE functions into your applications.

Turbo POWER TOOLS and Turbo ASYNCH sell for $99.95 each, and they may be ordered directly from Blaise Computing Inc. To order, call (415) 540-5441.

Turbo POWER TOOLS™

Turbo POWER TOOLS is a sleek new series of procedures designed specifically to complement Turbo Pascal on IBM and compatible computers. Every component in Turbo POWER TOOLS is precision engineered to give you fluid and responsive handling, with all the options you need packed into its clean lines. High performance and full instrumentation, including...

◆ Extensive string handling to complement the powerful Turbo Pascal functions. ◆ Screen support and window management, giving you fast direct access to the screen without using BIOS calls. ◆ Access to BIOS and DOS services, including DOS 3.0 and the IBM AT. ◆ Full program control by allowing you to execute any other program from within your Turbo Pascal application. ◆ Interrupt service routines written entirely in Turbo Pascal. Assembly code is not required even to service hardware interrupts like the keyboard or clock.

Using Turbo POWER TOOLS, you can now "filter" the keyboard or even DOS, and create your own "sidekickable" applications.

YES, send me the Best for the Best! Enclosed is _____ for
☐ Turbo ASYNCH ☐ Turbo POWER TOOLS. (CA residents add 6½% Sales Tax. All orders add $6.00 for shipping.)

Name: _____ Phone: _____

Shipping Address: _____ State: _____ Zip: _____

City: _____ Exp. Date: _____

VISA or MC #: _____

Turbo Pascal is a trademark of Borland International. Turbo POWER TOOLS, Turbo ASYNCH and PERFORMANCE PACKAGE are trademarks of Blaise Computing Inc. IBM is a registered trademark of International Business Machines Corporation. MS-DOS is a trademark of Microsoft Corporation.

BLAISE COMPUTING INC.

BLAISE

watch us!

◆ 2034 BLAKE STREET
◆ BERKELEY, CA 94704
◆ (415) 540-5441

Circle **no. 159** on reader service card.

Standard (DES) as the encryption system to be used by all federal agencies (except for those enciphering data classified under any of the national security acts). The standard is available in a government publication and also in several books. The DES was intended to be implemented only in hardware, probably because its designers did not want users to make changes to its internal tables. DES has been implemented in software, however, and is available in several microcomputer products (such as Borland's SuperKey or IBM's Data Encoder).

The DES is a product cipher using 16 stages of permutation and substitution on blocks of 64 bits each. The permutation tables are fixed, and the substitutions are determined by bits from a 56-bit key and the message block. This short key has caused some experts to question the security of DES. Controversy also exists regarding the involvement of the National Security Agency in parts of the DES design. The issues are interesting but beyond the scope of this article.

Because DES was intended for hardware implementation, it is relatively slow in software. Software implementations of DES are challenging because of the bit manipulation required in the key scheduling and permutation routines of the algorithm. Some implementations gain speed at the expense of code size by using large, precomputed tables.

The public-key cipher is an interesting new development that shows potential for making other encryption systems obsolete. It takes its name from the fact that the key information is divided into two parts, one of which can be made public. A person with the public key can encipher messages, but only one with the private key can decipher them. All public-key systems rely on the existence of certain functions for which the inverse is very difficult to compute without the information in the private key. These schemes do not appear to be practical for microcomputers—at least for 8-bit machines—if their strength is to be exploited fully. One variety of the public-key system (the knapsack) has been broken by solution of its enciphering function, but this is no reflection on other systems, such as the RSA scheme, which use different enciphering functions. All public-key systems proposed to date require heavy computation, such as the exponentiation and division of very large numbers (200 decimal digits for the RSA scheme). On the other hand, a public-key system that worked at only 10 bytes/second might be useful if all we are sending are the keys for some other system, such as the DES.

Some Random Thoughts

- Must we operate on blocks instead of bytes? Block ciphers seem stronger because they allow for permutation. On the other hand, they make life difficult when file size is not an integral multiple of the block size.
- Can we make a file encryption system OS-independent? This is related to the question above on blocks vs. bits. How do we define the end of file if the plaintext is ASCII and the ciphertext can be any 8-bit value?
- Can we find an efficient way to generate and store a random key for the infinite-key system? Hardware random-number generators are not hard to build, but would they be of any use?
- Bit fiddling is expensive. Can it be avoided and still leave a secure system?
- No file-encryption system can erase a file logically and be considered secure. The information can be recovered until it is overwritten. Overwriting files adds to processing time. I am informed that it is possible to reliably extract information even from sectors that have been overwritten. Is this so? If it is, what is the solution?
- How do we integrate encryption systems into different tools? Should a telecommunications program encrypt data transparently if the correspondent is compatible? What about an editor-encryption system wherein plaintext would never exist on the disk, only on the screen? How would we manage to encipher/decipher text as we scroll through it and make changes and still get acceptable performance?
- By their nature, encryption schemes are difficult to test. What test might we subject a system to that would increase our confidence in it?

Note

1. Claude Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal (Oct. 1949): 656—715.*

DDJ

From the Data Library

Listings One and Two, page 66, are the 68000 versions of the permutation routines described in Z80 code in the article "Designing a File Encryption System" in the August 1984 issue of *DDJ*. Permf performs the forward permutation of the bits in a 256-bit block as specified by a table of bytes. Permg performs the inverse permutation.

For example, if the permutation table has the values

```
1 15 115 57 . . . 0
```

then the forward permutation means to put the 1st bit of the block in the 0th place, the 15th bit in the 1st place, the 115th bit in the 2nd place, and so on until the 0th bit goes in the 255th place.

The inverse permutation with the same table means to place the 0th bit of the block in the 1st place, the 1st bit in the 15th place, the 2nd bit in the 115th place, and so on until the 255th bit goes in the 0th place.

In the original cryptographic use, the permutation table was assumed to be cycled to its next permutation after the encryption of each block. I will upload the cycle routine that does this fairly soon.

The routines address bits in the block by deriving a bit index from the byte value of the permutation table. The upper five bits of that value index to the particular byte in the block, and the lower three bits then index to the particular bit within that byte.

The routines run about three times faster than the Z80 versions.

(Listings begin on page 66.)

Vote for your favorite feature/article.
Circle Reader Service **No. 1.**

HOW PEOPLE WITH COMMON INTERESTS FIND AN INTERESTING COMMON GROUND.

Presenting CompuServe Forums. Where people from all over get together, without even leaving home.

Now thanks to CompuServe Forums, computer owners are sharing common interests by talking to each other through their computer keyboards. ***Software users, computer enthusiasts, ham operators, french cooks, fire fighters, science fiction lovers and other special interest groups*** are already in touch, online.

Because when you subscribe to CompuServe, you're able to reach people who want to talk about the things you do. As many people as you like. For as long as you like. Whenever you wish.

Join a conversation already in progress or start one on your own. Ask questions. And get answers.

All it takes is a modem, most any personal computer and CompuServe.

Forum members across the country are as close as a local phone call.

You can go online with just a local call in most major metropolitan areas. And normal usage fees for weekday nights and weekends are just 10¢ a minute

Of special interest to all Forum participants is software that's FREE for the taking.

Public domain software. For all sorts of activities, from games to business programs. And it's just as easy to copy a piece of software as it is to participate in a Forum.

Become a CompuServe subscriber and get a $25 Usage Credit to start you off.

Becoming a subscriber is as easy as contacting your local computer dealer. Or you can call us and order direct. Suggested retail price is $39.95.

And if you'd want more information about CompuServe, we'll be happy to send you a free brochure. Because with all that CompuServe offers—we think it's in your best interest.

CompuServe®

Information Services, P.O. Box 20212,
5000 Arlington Centre Blvd., Columbus, OH 43220

800-848-8199

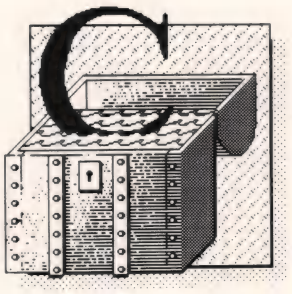
In Ohio, call 614-457-0802

An H&R Block Company

Circle **no. 237** on reader service card.

C CHEST

Sort—A General-Purpose Sorting Program



As tax time rolls around again (I'm writing this column in March), the dreaded task of organizing my tax records rears its ugly head. The year before last I used dBASE to do this organizing, and last year I used Lotus 1-2-3. Neither program is really satisfactory. It's too hard to enter data using dBASE, and the Lotus spreadsheet does just that, spread out all over my dining room table. Both programs are designed to do much more than needed anyway. All I want to do is create a normal file, with a normal editor, in which each line represents a deductible item. The line is split up into several fields (date, category, description, amount, check number), and the entire file must be sorted first by category and then by date (that is, all lines for a single category have to be grouped and the lines within the category need to be sorted by date).

So, my solution to this problem was to dust off an old sort utility and modify it so that it could handle files larger than the amount of available memory. This month, I'll discuss this program, which is called sort.

Sort sorts the lines of an ASCII file (or collection of files)—that is, the individual lines are rearranged but the words on a line aren't changed. Its command-line syntax is:

```
sort [-options] [file . . .]
```

If no files are given on the command line, standard input is used. If several files are listed, they are treated as if

by Allen Holub

they were one big file and then sorted. It's as if sort merged them all together into a single file and then sorted that single file.

Various command-line options are supported (see Table 1, page 24). These are:

-b—Ignore leading white space (blanks, tabs, form-feeds, and so on). If you're sorting on fields (see below), then white space following the field delimiter is ignored (even if the field delimiter itself is a space or tab).

-d—Sort in dictionary order. That is, all characters except letters and numbers are ignored for the purposes of comparison—for example, *hand puppet* will be between *handmade* and *handsaw* in the output.

-f—Fold uppercase letters into lowercase before comparing. Normally uppercase letters have a higher value than lowercase—*foo* will follow *Foo* in the output file.

-n—This flag is treated in a very different way from the way the Unix *sort* utility treats it. Here, if two numbers (with an optional leading - sign) appear in the same position on two lines, they are treated as a single number rather than as a collection of ASCII characters. For example, the list

```
 2
 1
20
10
```

will normally be sorted as a collection of ASCII characters, yielding

```
 1
10
 2
20
```

If *-n* is given on the command line, they'll be sorted as

```
 1
 2
10
20
```

Numbers are treated specially wherever they're found, even if they're imbedded in a word, so the list

```
word2
word1
word20
word10
```

will sort to

```
word1
word2
word10
word20
```

if *-n* is specified. Note, though, that the numbers have to be at the same relative place on the line, so

```
word  1
word  2
word 10
word 20
```

sorts to

```
word 10
word 20
words 1
words 2
```

unless *-d* is also specified.

Note that a version of *stoi( )*, the number-processing routine used for numeric processing, was originally published in the May 1985 C Chest. (See Listing Two, page 83.) The version given here differs from the original in that it's been scaled down to accept only decimal numbers (the original accepted hex and octal numbers too).

-t<c>—The single character <c>

C Programmers! First database written exclusively for C is also royalty free

"If you are looking for a sophisticated C Programmer's database, **db_VISTA™** is it..."
Dave Schmitt, President of Lattice, Inc.

Designed exclusively for C, **db_VISTA™** is a royalty-free programmers DBMS. Both single and multi-user versions let you take full advantage of C, through ease of use, portability and efficiency.

Written in C for C Programmers

All functions use C conventions so you will find db_VISTA easy to learn. db_VISTA operates on most popular computers, and because it is written in C it can easily be ported to most computers.

Royalty-Free, You only pay once

Whether you're developing applications for a few customers, or for thousands, the price of db_VISTA is the same. If you are currently paying royalties for a competitor's database, consider switching to db_VISTA and say goodbye to royalties. To help you make the change over to db_VISTA, file transfer utilites are available for dBASE, R:base and ASCII files.

More from your database applications with source code

Source code includes all db_ VISTA libraries and utilities.

1. Recompile our run-time libraries utilizing non-standard compiler options.
2. Create a debugging library including a function traceback by activating preprocessor commands embedded in the source code.

Multi-user and LAN capability

Information often needs to be shared. db_VISTA has multi-user capability and supports simultaneous users in either multi-tasking or local area networking environments, allowing the same C applications to run under UNIX and MS-DOS.

Faster execution without data redundancy

Less data redundancy means reducing disk storage requirements and maximizing data access performance. A customer evaluating a leading competitor's product prior to purchasing db_VISTA benchmarked db_VISTA's retrieval time to be 276% faster than a leading competitor.

Complete documentation included

User manual contains 193 pages, 8 diagrams, 10 tables, appendices, an extensive index, plus a database application example. 9 chapters with complete instructions.

Introducing db_QUERY™

With db_QUERY you can ask more of your database, db_QUERY is a linkable, SQL-based ad hoc query and report writing facility. It's also royalty-free.

30 day Money-Back Guarantee

We wish to give you the opportunity to try db_VISTA for 30 days in your development environment and if not satisfied return it for a full refund.

Special Offer

Free Falcon hard disk controller with purchase of db_VISTA Multi-user with source.*

Price Schedule

	db_ VISTA	db_ QUERY
Single-user	$195	$195
Single-user with Source	$495	$495
Multi-user	$495	$495
Multi-user with Source	$990	$990

Free 90 days application development support
All software not copy protected.

Call Toll Free Today!

To order or for information, call TOLL FREE 1-800-843-3313, at the tone touch 700-992 or 206-747-5570.
VISA and MASTERCARD Accepted

Read what others say about db_VISTA ..

"If you are looking for a sophisticated C programmers database, db_VISTA is it. In either a single or multi-user environment, db_VISTA lets you easily build complex databases with many interconnected record types. The multi-user implementation handles data efficiently with a LAN and Raima's customer support and documentation is excellent. Source code availability and a royalty-free run-time is a big plus."
Dave Schmitt, President Lattice, Inc.

"Not 'yet another user-friendly database', it is a DBMS aimed at the technical C programmer instead of the non-technical end-user".
Hal Schoolcraft, Data Based Advisor March, 1985

"On the whole, I have found db_VISTA easy to use, very fast with a key find, and powerful enough for any DBMS use I can imagine on a microcomputer".
Michael Wilson, Computer Language September, 1985

db_VISTA Version 2.11 Database Mangement System for C

Database Record and File Sizes
- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- No limit on number of records or set types
- Maximum file size limited only by available disk storage
- Maximum of 255 index and data files

Keys and Sets
- Key lengths may be a maximum of 246 bytes
- No limit on maximum number of key fields per record - any or all fields may be keys with the option of making each key unique or duplicate
- No limit on maximum number of fields per record, sets per database, or sort fields per set
- No limit on maximum number of member record types per set

Utilities
- Database definition language processor
- Interactive database access utility
- Database consistency check utility
- Database initialization utility
- Multi-user file locks clear utility
- Key file build utility
- Data field alignment check utility
- Database dictionary print utility
- Key file dump utility
- ASCII file import and export utility

Features
- Multi-user support allows flexibility to run on local area networks
- File structure is based on the B-tree indexing method and the network database model
- Run-time size is variable - will run in as little as 64K, recommended RAM size is 256K
- Transaction processing assures multi-user database consistency
- File locking support provides read and write locks on shared databases
- SQL based db_QUERY is linkable and royalty free
- Operating system support for MS-DOS, PC-DOS, Unix, Xenix or Macintosh
- C compiler support for Lattice, Microsoft, DeSmet, Aztec, Computer Innovations, Xenix and Unix
- File transfer utilities for ASCII, dBASE and R:base optional

Independent Benchmark Results

Eleven key retrieval tests on sequentially and randomly created key files. Benchmark procedure adapted from "Benchmarking Database Systems: A Systematic Approach" by Bitton, DeWitt, and Turbyfill, December, 1983

Total Retrieval Time of 11 Tests	
db_VISTA	:671.24
Leading competitor	:1,856.43

RAIMA™ CORPORATION
12201 S.E. Tenth Street
Bellevue, WA 98005 USA
(206) 747-5570
Telex: 9103330300 BCN RIVERTON

1 (800) 843-3313
at the tone touch 700-992

MasterCard VISA

Circle **no. 206** on reader service card.

*Limited offer available to end user purchases directly from Raima Corporation

(continued from page 22)

specifies a field separator character (the default is a tab, thus the *t*). For example, −*t*, tells sort to use a comma as a field separator.

−p<num>—Sort field <*num*> only where fields are delimited by the character specified with the −*t* argument. The leftmost field is field 1. For example, the file

ant, bat, cow
bat, cow, ant
cow, ant, bat

when sorted with the command line

sort −t, −p2 file

will yield

cow, ant, bat
ant, bat, cow
bat, cow, ant

Only the second field, rather than the entire line, is looked at by sort.

−s<num>—Specify a secondary key. When sorting fields, if the contents of the fields specified by the primary key are the same, then the secondary field is used to resolve differences. For example, the file

ant, cow, cow
bat, cow, ant
cow, ant, bat

when sorted with the command line

sort −t, −p2 −s3 file

will yield

cow, ant, bat
bat, cow, ant
ant, cow, cow

Here, the last two lines both have *cow* in the primary sort field (field 2), so they are ordered depending on what sort finds in the secondary sort field (field 3). Given a file containing lines of the form

<date>, <category>,
<other stuff>

my tax preparation can be done with the command line

sort −n −t, −p2 −s1 ledger
>outfile

Here the −*n* causes the dates to be sorted numerically, a comma is used to separate fields (−*t,*), the primary sort field is the category (−*p2*), and the secondary sort field is the date (−*s1*). The other fields on the line are ignored. The default primary field is *1* (the leftmost) if −*s* but no −*p* is given on the command line.

−r—Do a reverse sort (sort in descending rather than ascending order).

−T<str>—The <*str*> is prefixed to all intermediate file names. Intermediate files are usually called merge.1, merge.2, and so on. If you say

sort −T/tmp/ file

then the temporary files will be called /tmp/merge.1, /tmp/merge.2, and so on. Remember to put the trailing slash on the string if you're specifying a directory name (as compared to a RAMdisk designator or whatever).

−u—Delete duplicate lines in the output. After the file is sorted, only one of a series of identical lines is output.

−b	ignore leading white space (blanks)
−d	sort in dictionary order
−f	fold uppercase into lowercase
−n	sort numbers by numeric value
−p<num>	use field <num> as primary key
−r	do a reverse sort
−s<num>	use field <num> as secondary key
−t<c>	use <c> to separate fields
−T<str>	prepend <str> to temp file names
−u	delete duplicate lines in output

Table 1: *Sort command-line options*

```
ssort(array, nel, elsize, cmp)
char        *array;    /* Pointer to array being sorted */
int         nel;       /* Number of elements in array */
int         elsize;    /* Size of one element in bytes */
int         (*cmp)( ); /* Pointer to a comparison function */
```

Ssort( ) sorts the array using a shell sort. *Cmp* is a pointer to a comparison function that acts like *strcmp( )* does. *Cmp(a,b)* must return a negative number if *a*<*b*, zero if *a*==*b*, and a positive number if *a*>*b*. It is passed pointers to two array elements. *Argv* can be sorted with

```
acmp(a, b)
char        **a, **b;
{
            return( strcmp(*a, *b) );
}

main( argc, argv )
int         argc;
char        **argv;
{
            ssort( argv, argc, sizeof(*argv), acmp );
}
```

Table 2: *Calling syntax to* ssort

Sorting Large Files

Two general-purpose sort routines have been printed in this column: a quicksort routine (*qsort( )*) was printed in April 1985 and a shell sort

MULTIPLE CHOICE

TECH PC

MULTIUSER SYSTEM

FEATURES: TECH PC TURBO QUAD BUSINESS SYSTEM . . . . . .Starting from $5999

Tech Turbo PC/XT base unit in portable or desktop configuration with 640K, multiple serial ports, 20 megabyte hard disk, three Tech PC terminals, connecting cables, and networking software.

Separate NEC V20 8088 Intel compatible 8 MHz CPU and up to 768K RAM for each terminal on the system.

Two fully functional serial ports per terminal.

Four users expandable to 32 users over dumb terminals or PC's with terminal emulation software provided with the system. Capacity for unlimited number of local printers.

Full support for multitasking multiterminal use with print spooling for multiple printers, background monitoring of the system, dial up bulletin board support, password protection, and file/record locking supporting PC network protocol.

System support all popular software such as Wordstar, dBaseIII, Lotus 123, Multimate, etc.

FEATURES: TECH PC TURBO TRIAD MULTIUSER . . . . . .Starting from $2599

Tech Turbo PC/XT base unit with 640K, and two 360K disk drives, 20 megabyte hard disk.

Separate Intel 80188 microprocessor running at 8 MHz and 512K for each terminal.

Two high resolution monitors, two Selectric style Hi-Tek Keyboards, 50 feet of shielded cable to separate the two stations.

System expandable to three terminals.

System supports up to six printers.

Full support for multitasking multiterminal use with print spooling for multiple printers, background monitoring of the system, dial up bulletin board support, password protection, and file/record locking supporting PC network protocol.

System supports all popular software such as Wordstar, dBaseIII, Lotus 123, Multimate, etc.

FEATURES: TECH PC TWIN MULTIUSER . . . . . .Starting from $1699

Tech Turbo PC/XT base unit with 640K, and two 360K disk drives.

Two high resolution monitors, two Selectric style Hi-Tek keyboards, 50 feet of shielded cable to separate the two stations.

System supports up to six printers.

Full software support with multi-level file security, electronic message facility to send and receive messages between users, password logon system, and system operator command level.

System supports all popular software such as Wordstar, dBaseIII, Lotus 123, Multimate, etc.

FEATURES: TECH PC QUAD BUSINESS SYSTEM . . . . . .Starting from $4499

Tech Turbo PC/AT base unit in portable or desktop configuration with 512K, multiple serial ports, 20 megabyte high speed hard disk, three Tech PC terminals, connecting cables, and networking software.

Four users expandable to eight users over dumb terminals or PC's with terminal emulation software provided with the system.

Capacity for up to 16 printers at remote sites with up to 6 local printers attached to the main unit.

Each user can access 384K or more of RAM with memory expansion boards.

Full support for multitasking multiterminal use with print spooling for central or terminal printing, background monitoring of the system, dial up bulletin board support, password protection, and file/record locking using a general-purpose ENQ/DEQ mechanism callable from user applications.

System supports all popular software such as Wordstar, dBaseIII, Lotus 123, Multimate, etc.

THIRD PARTY MAINTENANCE AVAILABLE THROUGH MOHAWK DATA SCIENCES.

Users only circle no. 279 on reader service card. Dealers only circle no. 245 on reader service card.

TECH PC

TECH PERSONAL COMPUTERS
2131 South Hathaway, Santa Ana, California 92705

714/754-1170

TELEX: 272006 Answer Back-TECH **FAX: 714/556-8325**

PLEASE ALLOW ONE WEEK FOR SHIPPING

VISA, MASTERCHARGE

C CHEST
(continued from page 24)

(*ssort( )*) was printed in March 1986. Both algorithms were discussed in the April 1985 column. The *qsort( )* subroutine is also included in many compilers' I/O libraries. All these routines have an identical calling syntax so they can be used interchangeably.

These sort routines all have one major limitation. The entire array being sorted has to be in memory at once. Sometimes you need to sort files that are larger than the amount of available memory, however. This limitation is circumvented by using temporary merge files. Sort reads as much input as it can, sorts that input, and then writes the sorted lines to a temporary file. It continues in this manner until the input is exhausted, creating a bunch of temporary files, one for each pass. The program then merges the temporary files, writing the results to standard output. Finally, the temporaries are deleted.

The process is best illustrated with an example. Say that at most three input lines can be held in memory and a file containing

```
good
every
boy
favor
deserves
```

has to be sorted. Sort reads in the first three lines, sorts them, and then creates a temporary file. It then repeats the process with the remaining two lines. The temporary files look like

```
MERGE.1        MERGE.2

boy            deserves
every          favor
good
```

They are both sorted. The program now merges the files. It reads in the first line of each merge file

```
boy
deserves
```

outputs the lesser line (*boy*) and replaces it with the next line from the temporary file that originally contained *boy* (merge.1):

```
every
deserves
```

The program now repeats the same process on the two current lines, outputting *deserves* and replacing it with the next line from merge.2:

```
every
favor
```

every is output and replaced by the last line in merge.1:

```
good
favor
```

favor is now output. Because merge.2 is now empty, no input is fetched and the list contains the single word *good*, which is output in turn. Because all merge files are now empty, the program ends.

The same process can be used when there's more than one merge file. The program reads in one line

ATRON BUGBUSTERS GREASE BORLAND LIGHTNING

"If I were starting a software company again, from scratch, Atron's AT PROBE™ would be among my very first investments. Without Atron's hardware-assisted, software debugging technology, the flash of Turbo Lightning™ would be a light-year away." ***Philippe Kahn, President, Borland***

HOW BORLAND DOES SO MUCH, SO WELL, SO FAST

We asked Borland International president Philippe Kahn to share his secrets for rapidly taking a good idea and turning it into rock-solid reality. How does the Borland team do so much, so well, so fast?

He begins, "I remember when Atron used the June 24, 1985 *Wall Street Journal* chart of top-selling software in an ad." [Note: At that time, seven of the top ten software packages were created by Atron customers; it's now now nine out of ten.] "SideKick was number four, and I let Atron quote me in saying that there wouldn't have been a SideKick without Atron's hardware-assisted debuggers.

"You might say lightning has literally struck again. Turbo Lightning made number four on *SoftSel's Hotlist* within weeks of its introduction! And again, I say we couldn't have done it without Atron debugging technology.

"Cleverly written code is, by definition tight, recursive, and terribly complex," he continues. "Without the ability to externally track the execution of this code, competent debugging becomes very nearly impossible."

Concludes Philippe, "And after Turbo Lightning was solid and reliable, Atron tuning software turned our Probes into performance analyzers. How do you think we *greased* our lightning?"

Philippe, along with a couple million or so of your satisfied customers, we say congratulations on yet another best-selling product. We can't wait to see what awesomely useful technology will come shooting out of Borland International next.

HOW BUGBUSTERS KEEP YOU FROM GETTING SLIMED

The AT PROBE is a circuit board that plugs into your PC/AT. It has an umbilical which plugs into the 80287 socket and monitors all 80286 activity.

Since AT PROBE can trace program execution in real time, and display the last 2048 memory cycles in symbolic or source-code form, you can easily answer the questions: "How did I get here?" and "What *are* those silly interrupts doing?"

It can solve *spooky* debugging problems. Like finding where your program overwrites memory or I/O - impossible with software debuggers.

You can even do source-level debugging in your favorite language, like C, Pascal or assembler. And after your application is debugged, the AT PROBE's performance measurement software can isolate performance bottlenecks.

Finally, the AT PROBE has its own 1-MByte of memory. Hidden and write-protected. How else could you develop that really large program, where the symbol table would otherwise take up most of memory.

LOOK AT IT THIS WAY.

History shows that non-Atron customers don't stand a very good chance of making the Top Ten list. Lightning *really does* have a way of striking twice!

The PC PROBE™ is $1595 and the AT PROBE is $2495. So call Atron today. You can be busting some really scarey bugs tomorrow. And maybe, just like Borland, you can also bust some records.

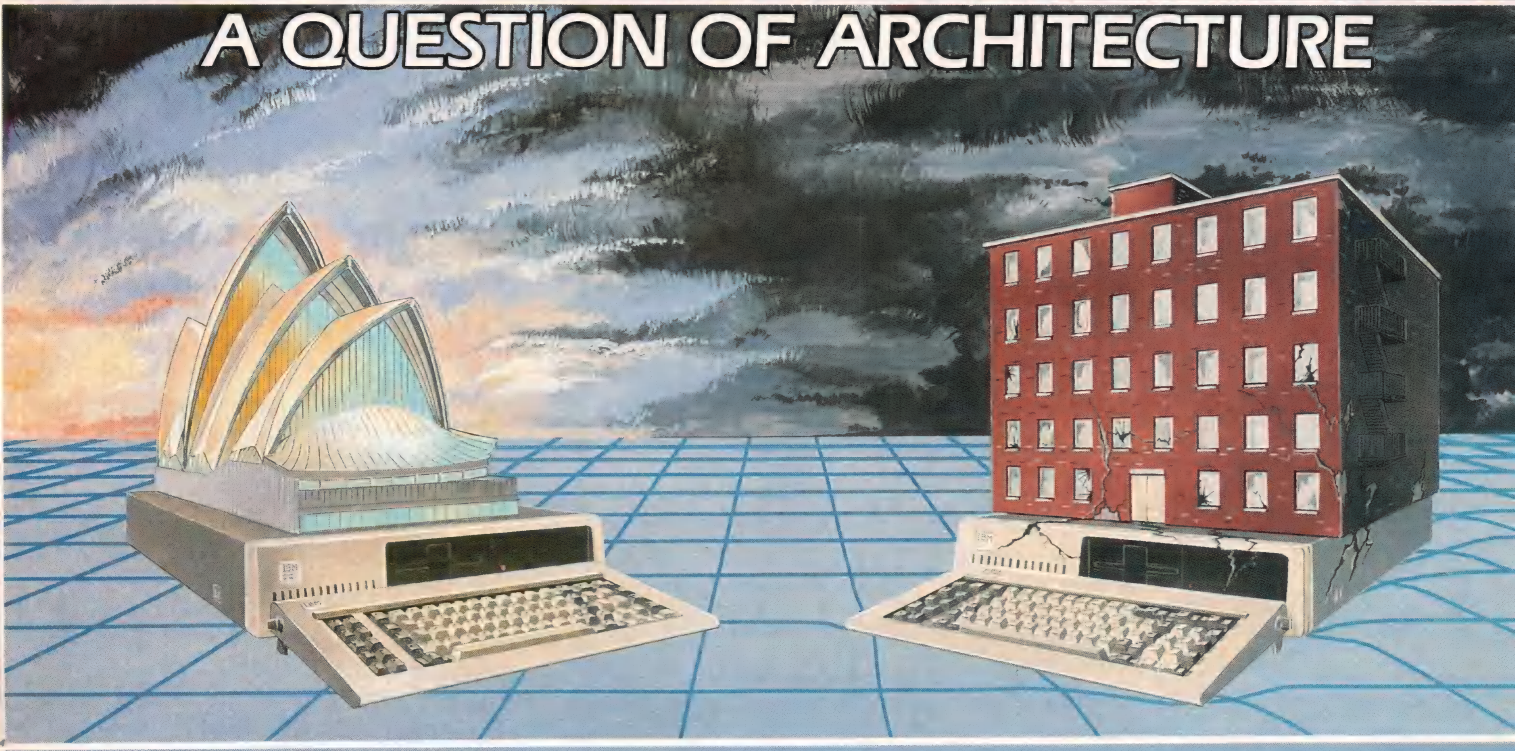
Atron
THE DEBUGGER COMPANY
20665 Fourth Street • Saratoga, CA 95070 • 408/741-5900

Copyright © 1985 by Atron Corp. PC PROBE™ and AT PROBE™ Atron. SideKick™ and Turbo Lightning™ Borland International, Inc., Adv. by TRBA, 408/258-2708.

Circle no. 216 on reader service card.

QNX vs UNIX

A QUESTION OF ARCHITECTURE



What do QNX and UNIX have to do with architectural design?

The design determines the environment in which you and your applications must survive. If the shear weight of the UNIX operating system brings the PC to its knees, all applications running under it will suffer. Unix was conceived more than a decade and a half ago and the product today is the result of modifications, additions and patches by hundreds of programmers. The result is a large and convoluted piece of software which needs the resources of an AT or more.

QNX's superb performance and compact size is the result of one dedicated design team with a common purpose, and complete understanding of both the software and the environment in which it must run. It runs quickly and efficiently on PC's and soars on an AT. Unlike Unix, QNX is capable of real time performance and is the undisputed choice for real time process control, and office systems. You can buy an OS that offers you a 1 to 3 user dead end on an AT, OR, you can consider QNX which allows you anywhere from 1 to 10 users on both PC's and AT's. And we don't stop there. Unlike other Unix-type systems for PC's, QNX is also a networked operating system. Not a patch-on network, but a fully integrated networking system for up to 255 micros. QNX allows you to start with a single machine and grow if and as required. There are no dedicated file servers and you can attach terminals (users) to any machine. To choose a solution which ignores networking, is closing the door on your future.

Everyone is talking about Unix like systems, but no one wants to abandon the tremendous amount of DOS software available. QNX does not force you to make that decision. You can run either PC DOS 2.1 or 3.1™ as one of QNX's many tasks. (DOS File compatibility and DOS development tools are also available). Don't misunderstand us. We at Quantum have a great deal of respect for Unix. It was a major force in moving operating systems out of the 1960's and into the 70's. QNX however, was designed in the 80's and will be a driving force of the 1990's. **Over 20,000 systems have been sold since 1982.**

Quantum strongly believes that there are good reasons for buying QNX, DOS and Unix. If you want more than DOS and a working alternative to PC Unix, give us a call and we will discuss your needs.

End-Users, VAR's, OEM's and software developers are invited to take the QNX challenge.

• **MULTI-USER:**	**-10 serial terminals per PC, AT.**
• **MULTI-TASKING:**	**-40 tasks per PC, AT.**
• **NETWORKING:**	**-255 machines.** **-up to 10,000 tasks and 2000 users/network.** **-2.5 Megabit token ring.**
• **REAL TIME:**	**-2800 task switches/sec (AT).**
• **MESSAGE PASSING:**	**-Intertask communication between any of 1000's of tasks on any machine.**
• **MEMORY:**	**-88K to 110K for QNX.**
• **PC DOS:**	**-Executes as a task under QNX.**
• **C Compiler:**	**-Standard Kernighan and Ritchie.**
• **Flexibility:**	**-Single machine or networked. One to thousands of users. Full resource sharing of disks and devices on all machines.**
• **Support:**	**-Online update system allows downloading of new releases over the phone.** **-Technical support hot line.**
• **COST:**	**-From US $450 for base system.** **-Call for runtime prices.**
• **HARDWARE SUPPORT:**	**-IBM PC, XT, AT™ (both real & protected) and compatibles.**

QUANTUM SOFTWARE SYSTEMS LTD.

Moodie Drive HiTech Park 215 Stafford Rd. Ottawa, Ontario, Canada. K2H 9C1 Phone (613) 726-1893

IBM PC, AT, XT AND PC-DOS ARE REG. TM OF INTERNATIONAL BUSINESS MACHINES CORP. UNIX IS A REG. TM OF AT&T BELL LABS

Circle **no. 144** on reader service card.

Make your PC or AT into a COMMUNICATING WORKSTATION for only $85

Use *ZAP*, the Communications System for Technical Users
COMPLETE Communications for PROGRAMMING and ENGINEERING

EMULATION of graphics and smart terminals is combined with the ability to TRANSFER files reliably, CAPTURE interactive sessions, and transmit MESSAGES while also being able to swap between your mini or mainframe session and your PC application. SUSPEND a line to run a PC application. Reconfigure features to fit the communications parameters and keyboard requirements of the host computer software. Complete technical documentation helps you understand and fit ZAP to your style.

HIGHLIGHTS OF *ZAP:*

- Emulate TEKtronix 4010/14 and DEC VT 100, 102, 52 including variable rows and columns, windows, full graphics, and smooth scrolling.
- Reliable ***file transfer*** to/from any mainframes and PCs including KERMIT and XMODEM protocols plus you get a full copy of KERMIT. Transfer ***speeds*** ranging from 50 to 38,400 BAUD. Session control include ***printer dumps*** and ***save to disk***.
- **MACRO and Installation files** ("scripts") controllable by you.
- EMACS, EDT and VI "Script" files are included. ZAP is also used with other popular software including graphics products like DISSPLA and SAS/GRAPH.
- **CONFIGURABLE** to communications, terminal features on the "other end"; 1, 2 stop bits; 5, 6, 7 or 8 data bits; parity of odd, even, none, mark and space; remap all keys including the numeric pad and standard keyboard, set any "virtual" screen size.
- Full **PC/MSDOS** access to run any command or program that will fit in your systems memory. ZAP takes less than 64K.
- 9 Comm ports are supported by ZAP. Plus full color in text and graphics make use of the IBM color, EGA cards, or Hercules Monochrome.

ONLY $85

Full refund if not satisfied during first 30 days.

Solution Systems™

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle **no. 148** on reader service card.

The C Programmer's Assistant

C TOOLSET™

UNIX-like Utilities for Managing C Source Code

No C Programmer should be without their assistant – C ToolSet from Solution Systems. The package consists of several utilities designed to help make C programming tasks easier.

C ToolSet (formerly C Helper) includes:

DIFF – Compares text files on a line-by-line basis or use CMP for byte-by-byte – indispensable for showing changes among versions of a program under development. So "intelligent" it stays in synch even when you add 100 lines.

GREP – Regular expression searches – ideal for finding a procedural call or a variable definition amid a large number of header and source files.

FCHART – Traces the flow of control between the large modules of a program.

PP (C Beautifier) – Formats C program files so they are easier to read.

XREF (CCREF) – Cross references variables from a program.

Available For MS-DOS – $95

ONLY $95
Source Code Included

Full refund if not satisfied during first 30 days.

Solution Systems™

335 Washington St.
Norwell, MA 02062
617-659-1571
800-821-2492

Circle **no. 152** on reader service card.

(continued from page 26)

from each file, outputs the line having the lowest value, and then replaces that line with the next line from the appropriate merge file, continuing the process until all the merge files are empty.

The problem here is selecting the line having the smallest value. The simplest method is to keep an *argv*-like array of pointers to strings, where each string is one line from a merge file. The array is sorted on each pass, and the lowest element is output. Sorting the array each time isn't very efficient, though. You're constantly resorting an almost-sorted array. A better way to store the lines is in a data structure called a *heap*. A heap is an array that has the property that the Kth element is always less than the (2K)th and (2K+1)th elements. For example, *array[1]* is less than both *array[2]* and *array[3]*, *array[2]* is less than both *array[4]* and *array[5]*, *array[3]* is less than both *array[6]* and *array[7]*, and so on. You can look at a heap as a sort of binary tree, where the Kth element is the parent node and the (2K)th and (2K+1)th elements are the children. All sorted arrays are heaps; on the other hand, a heap is not necessarily a sorted array.

Heaps have two additional properties that are of use here: The first element is always the smallest element, and a new element can be inserted in the heap in O(logN) time, where *N* is the heap size.

Implementation Details

Sort is presented in Listing One, page 68. Various *#defines* are on lines 18–28. *MAXBUF* is the maximum line length (now 132 columns). Lines longer than this will be treated as if they were two lines. *MAXLINEC* (now 1024) is the maximum number of input lines than can be read before a merge file is created. *MAXTMP* (now 18) is the maximum number of temporary files that can be open at once. This number is limited by either your compiler or the operating system. Two file descriptors are needed for *stdout* and *stderr*, so I'm assuming here that 20 files may be open at one time. This assumption isn't true in some CP/M systems that only allow

eight files. In DOS, to have 20 file descriptors available, you must say

files=20

in your config.sys file. It will speed up your I/O to put

buffers=20

up there too.

The global variables on lines 36–45 are all set by *getargs( )*, depending on what it finds on the command line. (The source for *getargs( )* isn't in the listing; see below for availability.) The *Argtab* on lines 47–61 is also used by *getargs( )*. Global variables not concerned with command-line switches are declared on lines 70–88. *Options* is set to true by *main( )* if any of the command-line switches that affect the sort order are present. *Lines* and *Linec* are used in the same way as *argv* and *argc*. *Lines* holds pointers to the input lines read in during the initial sorting passes (not the merge passes). *Linec* is the number of valid lines in *Lines*. *Argv* and *Argc* are just global copies of the *argv* and *argc* used by *main( )*.

The *HEAP* structure defined on lines 80–85 is used to define the heap needed in merge-file processing. You need to remember two things about every heap entry—the contents of the current line in the merge file and the *FILE* pointer associated with that merge file. So you use a structure having two fields: *string* and *file*. The heap itself is an array, *MAXTMP* elements long, of pointers to *HEAP* structures. I chose to use an array of pointers rather than an array of structures because it takes much less time to exchange two pointers than it does to exchange two, rather large, structures. This pointer array is created using *malloc( )*, and it is pointed at by the global variable *Heap* (defined on line 87).

The routine *pheap( )* (lines 95–108) is a debugging routine that prints out the current heap contents. Note that if *DEBUG* isn't *#defined*, then *pheap( )* will be *#defined* as a null macro (one that expands to a null string). This way you don't have to put *#ifdef DEBUG/#endif* directives around all the calls to *pheap( )*.

Lines are read into memory by *gtext( )* (line 358f). It loads the input lines into an *argv*-like array of pointers to strings (*Lines*). The routines that call *gtext* don't know that it may be getting input from several files (as listed on the command line). *Gtext* [or rather *nextfile( )* (line 324f), which is called by *gtext( )*] takes care of all the *argv* processing needed to open and read the various files when appropriate.

Skipping forward, the initial sorting of the *Lines* array is done by the call to *ssort( )* on line 617. The calling syntax for *ssort( )* is shown in Table 2, page 24.

The beauty of passing a pointer to a subroutine becomes obvious when looking at a routine such as *ssort( )*. Not only can an array of pointers to strings be sorted with the routine shown in Table 2 but also a more complicated sort (such as the one required by my sorting program) can be performed by the same sort subroutine. Just pass it a pointer to a more complicated comparison function. The comparison function used here is actually several subroutines defined on lines 172–320. *Argvcmp( )* (line 172f) is used when

Lattice® Works

RPG COMPILER FOR IBM PC

The new Lattice RPG II compiler is ideally suited for creating commercial applications for MS-DOS. Allow your current RPG II programmers to be productive on MS-DOS.

The Lattice RPG II compiler is compatible with System III, System/34 and /36 RPG II compilers, it uses ASCII files and MS-DOS command language, plus has ISAM compatibility with dBASE III. $750.00 and no run time fees.

VERSION 3 OF THE LATTICE MS-DOS C COMPILER IS NOW AVAILABLE.

This is a major upgrade of the product and is available to registered users for a $45 update fee. Non-registered $60. The list price remains $500.

New compiler features include:

- *ANSI language constructs...*
  "unsigned" as a modifier
  "void" data type
  "enum" data type
  structure assignments, arguments, and returns
  argument type checking
- *Inline code 8087/80287 80186/80286*
- *Code generation*

The compiler also contains numerous improvements such as better aliasing algorithms, more efficient code generation, and more flexible segmentation. The library includes more than 200 new functions in the following categories:

- *ANSI/UNIX/XENIX compatibility*
- *Extended support for MS-DOS*
- *Extended support for networking, including file sharing, file locking, and I/O redirection*
- *Flexible error handling via user traps and exits*

The Library has also been re-engineered to produce much smaller executables.

LATTICE ANNOUNCES NEW DATA ENCRYPTION SOFTWARE

Now you can keep your confidential data confidential. Thanks to new SecretDisk, a new data encryption system for IBM PC, XT, AT and compatibles.

Utilizing the NBS Data Encryption Standard, SecretDisk provides complete security for salaries, customer lists, or other sensitive information stored on a floppy or hard disk. SecretDisk is loaded as a disk driver by MS-DOS. It creates new DOS drives (like D:) on floppy or hard disks where all data and programs are always fully encrypted.

SecretDisk is extremely easy to use. A password is entered when the system is booted, and protection can be switched on and off with a single password controlled command line. However, without the password, there is no way to access the encrypted files. $59.95.

Contact Lattice, to discuss your programming needs. Lattice provides C compilers and cross compilers for many environments including Tandy, Sony, Hewlett-Packard, Tandem, and IBM Mainframe. Corporate license agreements available.

Lattice

(312) 858-7950 TWX 910-291-2190
INTERNATIONAL SALES OFFICES;
Benelux: De Vooght. (32)-2-720-91-28.
Japan: Lifeboat Inc. (03) 293-4711
England: Roundhill. (0672) 54675
France: SFL (1) 46-66-11-55
Germany: (49) 7841/4500 (49) 8946/4613-290

Circle no. 101 on reader service card.

C CHEST
(continued from page 29)

no command-line switches that affect the sort order are specified; *qcmp( )* (line 180f) is called when command-line switches are specified. Both routines are passed pointers to string pointers (that is, the addresses of two elements of *Lines*) and both call a workhorse function to actually do the work. *Argvcmp( )* calls *strcmp( )* after stripping off one level of indirection; *qcmp( )* calls *qcmp1( )* (line 194f), which in turn calls *qcmp2* (line 228f). The former takes care of sort fields, whereas the latter does the actual comparisons—doing things such as skipping white space, mapping uppercase into lowercase characters, and so forth—as specified by command-line switches.

Once the lines are sorted, they are written out via the subroutine *outtext( )* (line 422f). *Outtext* will write to standard output if no merge files need to be created; otherwise it will write to a temporary file. It outputs the entire *Lines* array and deletes the memory used by the strings themselves (as compared to the memory used by the pointers to those strings).

As the last step in the sort, all the temporary files are merged together and the result sent to standard output. The actual merging is done in *merge( )* (line 542f). The routine *open_mergefiles( )* (line 455f) creates the heap, opens all the merge files, and reads the first line from each merge file into the heap. The heap is then initialized with the *ssort* call on line 546 (remember, a sorted array is a legal heap). The *while* loop on lines 548–566 prints the smallest element of the heap (the one pointed to by **Heap*) and then reads another line from the appropriate file. If there are no more lines to read, the file is closed and the heap is made smaller by copying the entire heap, overwriting the first entry, and decrementing *nfiles*. Finally, *reheap( )* is called to put the new entry at its proper place in the heap.

Reheap( ) (line 507f) reorders the heap in a manner similar to a binary-tree search (except that *reheap( )* isn't recursive). It compares the parent element to the two children and, if the parent is smaller than a child, transposes the two elements. The process is then repeated with the newly transposed child used as the parent node. This way the new entry percolates to its proper position in the heap. Unlike other "percolating" strategies, such as bubble sort, *reheap* is efficient, using at most $\log_2 N$ swaps, where N is the heap size rounded up to the nearest power of 2.

Limitations

There are practical limits on the amount of data that can be sorted. Because each merge file can contain at most 1,024 lines and 18 merge files are permitted, only 18,432 lines can be sorted (split up into as many input files as you like). Another consideration is the amount of space available to *malloc* (about 58K in my version of sort). Again, because this number limits the size of a merge file, there can be no more than 58K × 1,024 bytes in the combined source files (about a megabyte). Of course this last number will vary a little depending on the line lengths, but it's close. In practice, these limits haven't been a problem, but you may need to go to a large-model version of this program (to increase the amount of space available from *malloc*) or use a more sophisticated merge algorithm.

Availability

The source code for the entire program this month (including the *getargs( )*, *ssort( )*, and *stoi( )* routines) is available on CompuServe in the DDJ Forum. It's also available on an IBM PC-compatible disk for $25 from Software Engineering Consultants, P.O. Box 5679, Berkeley, CA 94705. The disk comes with an executable version and source code.

Getargs( ) and *stoi( )* were originally published in the May 1985 C Chest; *ssort( )* was published in the March 1986 C Chest. The source for all three routines is also available as part of the /Util utility program package. It costs $29.95 and is available from *DDJ*. (See the ad in the *DDJ* ad catalog in this issue, page 73.) **DDJ**

(Listings begin on page 68.)

Vote for your favorite feature/article.
Circle Reader Service **No. 2.**

SuperLinker+

The SLR SuperLinker Plus is 3 - 10 times faster than any other linker, and look at these features:

- link a full 64K output (COM, HEX, SPR or PRL)
- works with Microsoft Fortran, Basic, Cobol
- supports 32 character externals (SLR format)
- full drive/user support with alternate DU search
- supports 8 address spaces
- fill uninitialized spaces with 0 or FF
- global cross reference
- DSD80/SID compatible .SYM file
- manual overlays
- load map

requires Z80 CP/M 2.2 or greater 32K TPA

SLR Systems

1622 N. Main St., Butler, PA 16001
(800) 833-3061 (412) 282-0864
Telex 559215 SLR SYS

Circle **no. 78** on reader service card.

For you, good is just not good enough.

For serious programmers, good isn't good enough. You need the best ... the best tools, the best advice and the best support.

At Lifeboat, we've been selling to programmers since 1976, so we know quality when we see it. And we're committed to a full-service program that goes beyond just selling you the best software at competitive prices. Our expert staff can help you decide which programs are best for your needs and provide you with all the technical support you may require. You can rely on Lifeboat for the complete solution to your programming needs.

LANGUAGES

Lattice C *New 3.0 Version*

The best selling C Compiler has been upgraded to give you more functions and features. Lattice C 3.0 contains 200 new library functions, better code generation, support for new data types (void, enum, unsigned char, unsigned long), support for the 80186/80286 instruction set and the ability to generate in-line 8087/80287 instructions. Lattice C is the C Compiler for professional developers.

RUN/C—The C Interpreter *Upgraded Version*

Learn C the natural way with RUN/C. The user interface is similar to BASIC with easy familiar commands. The new 2.0 version of RUN/C comes with a full-screen editor and other enhancements.

RUN/C Professional *New*

All RUN/C's capabilities plus powerful features for program development. Load your favorite object libraries with RUN/C Professional. Contains a full-screen editor and source code debugging facilities.

Mark Williams C Programming System

A complete C development environment.

Pro Pascal

A truly standard Pascal. Produces fast, tight code with plenty of compile time options and simple one-line commands.

BetterBASIC *New Version*

Now you can program in BASIC and use the full memory of your PC, create structured programs using functions and procedures, make your own library modules and more. Now compatible with Microsoft BASIC.

LANGUAGE UTILITIES

Plink86 Plus

An overlay linkage editor for linking 8086/8088 object modules. Supports an unlimited size file, unlimited number of modules and up to 4095 hierarchical overlays stacked as many as 32 levels deep. Plink86 Plus contains new features for memory caching, library allocation, file merging and overlay reloading.

Pfix86 Plus

A symbolic and source level advanced debugger for programming professionals.

C-SPRITE

Symbolic debugging at both source level and machine level. Monitor your program by setting breakpoints, examining registers and variables, or single-stepping through code. Handle a set of commands as a macro. C-SPRITE and the Lattice C Compiler are a natural pair for program development and debugging.

BASTOC

A BASIC to C translator for the BASIC programmer who wants to upgrade to C.

EDITORS

LSE

A powerful yet inexpensive full-screen editor designed for efficiency and ease of use. Its speed is optimized by writing directly to video memory. The Lattice Screen Editor provides a multi-window environment, ability to reprogram keys, support for keyboard macros, an undo command, an error tracking mode, on-line help and more.

VEDIT Plus

A full-screen text editor for program development and word processing. It contains powerful features including use of macros, on-line help facility, paragraph formatting, and file comparison.

Pmate

The programmer's editor with an extensive macro command language. Compile in the background while you continue to edit files.

EMACS

Customizable editor including windowing, multi-tasking and special modes for C and Pascal.

FUNCTIONS

C-Food Smorgasbord

Library of time saving utility functions including a BCD package, an IBM PC BIOS interface, level 0 I/O functions, a terminal independence package and more.

Essential C Utility Library

Over 300 functions, with special attention given to screen handling, windows and business graphics. Source code is included.

PforCe *Brand New*

An optimized library of an impressive 400 plus functions and subsystems. Included is a window management system with overlapping or tiled windows and a database system with B-tree file structure. Several pre-coded screens including "Lotus" style are supplied. And there's much more! Routines for interrupt driven communications, background tasks, and string/table parsing, along with functions for field/screen editing and validation are all part of this superbly written and documented software. Complete source code is included.

The Greenleaf Comm Library

A library of over 120 communication routines. Contains functions to create interrupt driven routines or perform direct I/O to multiple Comm Ports. Its strengths are in asynchronous communications, interrupt mode, modem control, XMODEM, XON/XOFF and flow control. Interfaces with Lattice, Microsoft, Wizard, Desmet, CI86, Aztec and Mark Williams C compilers.

The Greenleaf Functions

A mature library of over 200 functions. Version 3.0 offers all new indexed documentation, with an abundance of examples. Source code included.

GRAPHICS and SCREEN DESIGN

GSS*CGI GRAPHICS *New*

The GSS Computer Graphics Interface is designed for creating high performance graphics-based applications. GSS*CGI speeds up application development and provides compatibility with a wide range of peripherals. It's the only CGI implementation that provides true device-independence for both raster and vector graphics. Language bindings are available to support development of GSS*CGI based applications using Lattice C, Microsoft C, FORTRAN, Pascal, BASIC Compiler and Macro Assembler. Products in the GSS*CGI GRAPHICS line include the GSS Graphics Development Toolkit, the GSS Kernel System, the GSS Plotting System and the GSS Metafile Interpreter.

Essential Graphics *New*

A brand new graphics library for C programmers with the emphasis placed on ease of use and portability. No royalties.

Multi-Halo

Library of over 200 graphics functions, supporting all of the popular graphics boards.

Panel

A powerful tool for interactive screen design.

For more information on these and other products in our complete line call:

1-800-847-7078 In NY: 914-332-1875

LIFEBOAT

The Full-Service Source for Programming Software.

INTERNATIONAL SALES OFFICES

Australia:
Fagan Microprocessor Assoc.
Phone: (61) 3699-9899
Canada: Scantel Systems
Phone: (416) 449-9252
England: Grey Matter, Ltd.
Phone: (44) 364-53499
Italy: Lifeboat Assoc., S.p.A.
Phone: (02) 656-841
Japan: Lifeboat Japan
Phone: (03) 293-4711
Spain: Micronet, S.A.
Phone: (34) 1-457-5056
The Netherlands:
GIGA Computer Products
Phone: (31) 10-771848
SCOS Automation BV
Phone: (31) 20-106922

Lifeboat
55 South Broadway
Tarrytown, NY 10591

Circle **no. 118** on reader service card.

The names of products listed are generally the trademarks of the sources of the products.

©1986 Lifeboat Associates

How to Fix Line Glitches

by Joe Marasco

Forward error correction deals with data-transmission errors by correcting them as the data are received.

Most telecommunications users are accustomed to getting error-free messages when moving blocks of data during file transfer operations. This situation differs greatly from normal bulletin board operations or from using a remote computer interactively. In these situations, most people don't even flinch when Ma Bell routes their call over barbed wire, and the ubiquitous ~ appears. Glitches on the line are almost taken for granted, and rightly so, because the "error" is obvious and immediately discounted. When transferring files, and in particular binary files, however, one bit error can be fatal, and more important, there is no easy way to detect it. It is for this reason that error-checking mechanisms are a fact of life in the telecommunications world.

The most common forms of error checking are the simple parity check and the CRC (cyclical redundancy code) check. *(For an extensive discussion of CRC techniques, see Terry Ritter's article, "The Great CRC Mystery," in the February 1986 issue of* DDJ.*—ed.)* In these schemes, one or more bits are computed when each block is sent; the block then contains the message bits and the appended computed bits. The receiver recomputes the check bit or bits using the message bits only and compares the result to the computed bits sent. If they match, an ACK is sent to the transmitter. If there is no match, a NAK is sent. The transmitter must then send the block again. CRCs provide a high degree of robustness for a small overhead.

In some applications, however, retransmissions are prohibitively expensive. When data is coming from very far away—say, from Saturn—and in large blocks, the time lag to resend the block can be significant. If the data has a real-time aspect to it, retransmission can in effect be worthless because, by the time it is evident that old data needs to be resent, new data may already be in the queue.

Forward error correction addresses these problems. The key idea is that by giving up some of the transmission bandwidth — sacrificing more message bits for overhead bits—we can not only detect the presence of an error but also correct the error at the receiver. The basic notion is that we never do a retransmission; we design the system subject to a known noise environment such that we can "fix" all "broken" packets when we get them.

Joe Marasco, Billybob Software, P.O. Box 363, Belmont, CA 94002-0363.

© 1986, Billybob Software. All rights reserved.

Single-Bit Errors

Let's demonstrate this principle with a concrete example using the simplest of these codes, the Hamming code.[1] The Hamming code enables you not only to detect the presence of one bit error in a block but also to locate its position and hence correct it. For now let's assume we are only concerned with single-bit errors.

For the sake of example, let us assume that we are going to transmit a 15-bit block. It turns out (we'll show you how later) that it takes 4 bits of "parity" check for a Hamming code for this block size, so we have only 11 bits of data left for the "message." Although this is a high overhead, it is purely a result of having such a small block; the situation improves radically as we go to larger blocks. We use a small message here for ease of exposition.

Because we are going to send 15 bits, we are going to have to localize the bit error to one of 15 positions. If we make a table of the 15 positions and their binary representations, we can add a third column that addresses the question of which one of the four parity checks should be applied to each bit. (See Table 1, page 33.)

Notice that the entries in the parity check column are unique, as well they should be. All we have done really is to say that if the "one bit" is in error, it will affect all the "positions" that have a "one bit," and so on. We can rewrite Table 1 by noting explicitly which bits or positions are governed by each parity check (C1, C2, C3, or C4), as shown in Table 2, page 33.

We're now almost ready to encode a message! The trick is to reserve bits 1, 2, 4, and 8 for the check bits, leaving 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, and 15 for the message bits. Let's also decide that the total number of bits governed by any

check must be even. Here's an arbitrary message:

```
 1   2 3   4 5 6 7   8 9 10 11 12 13 14 15
C1 C2 1 C3 0 1 1 C4 0  1  1  0  1  0  1
```

Now let's compute C1, which is governed by the parity of the bits shown in Table 2. Looking at the odd bits from 3 to 15, we count five 1s, so for the parity to be even, C1 must be a 1. Our message now looks like this:

```
1   2 3   4 5 6 7   8 9 10 11 12 13 14 15
1 C2 1 C3 0 1 1 C4 0  1  1  0  1  0  1
```

To compute C2, we do the same thing for the bits listed under C2 in Table 2. Looking at all the bits except for 2, which we are trying to find, we count six 1s, which is even. So C2 must be a 0. This yields:

```
1 2 3   4 5 6 7   8 9 10 11 12 13 14 15
1 0 1 C3 0 1 1 C4 0  1  1  0  1  0  1
```

You can do C3 and C4 by yourself. The final block, containing the message and check bits, should look like this:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 0 1 0 0 1 1 0 0  1  1  0  1  0  1
```

So the message sent would be "101001100110101."

What happens at the receiver? Clearly, if the message is received with no errors, each parity check will pass, and, by definition, we ascribe a 0 to a good parity check. When we make a number out of the four binary digits from the parity check, we get 0000 for the location of the error. This value is called the syndrome; a zero value for the syndrome means there were no errors.

Now let us suppose that an error is made in transmission. Suppose that the error occurs in the fifth bit sent. Our table then looks like the one shown in Table 3, right. Applying the parity checks as given in Table 3, one by one, we have the table shown in Table 4, right. And when we construct the syndrome (C4C3C2C1), we get

0101 −> 5 −> error is in the fifth bit

So now we know that all we have to do to correct the message is to invert the fifth bit, making it a 0 instead of a 1. The original message is recovered.

There is something very nice about the Hamming code and all forward error correction codes in general. Note that once the encoding is done, there is a total equality between the original message bits and the computed check bits. It doesn't matter at all if during the transmission a data bit or a check bit gets squashed—they are all on an equal footing. If a check bit gets reversed, it will be corrected; in effect the original message gets through OK and doesn't need "correcting."

Two-Bit Errors and More

What happens with this simple code when two bit errors occur? Try it! You'll find that 2-bit errors in the channel will cause the decoder to "miscorrect," introducing yet a third bit in error. This is indeed a sad state of affairs.

At the cost of one additional check bit we can improve the situation, however. Let the zeroth bit be the total parity after the other 15 bits have been encoded. Now we have an interesting situation. If the syndrome is zero and the parity bit is OK, we have a good message. If we have one bit error in the 15 bits, we have a nonzero syndrome and a bad parity bit; we can locate the bad bit as before. If just the parity bit itself gets reversed, the syndrome will be zero, so we'll know it's just the parity bit. Finally, and most important, if there are 2-bit errors anywhere, we will get a nonzero syndrome and a good parity bit. That is the unmistakable signature of a 2-bit error. So what we now have is much better: a code that corrects any single-bit error and detects all 2-bit errors.

To let you try this out, I've written a blatant hack. (See Listing One, page 84.) You input an 11-bit message, and the program computes and displays the completed 16-bit block that would be transmitted. You can then type in the "received" message, corrupting one or more of the bits, and the decoder will tell you which bit is in error and display the corrected message.

We've just scratched the surface. Hamming codes are the first step in doing forward error correction. You've

Position	Binary Representation	Parity Check
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	1, 2
4	0 1 0 0	3
5	0 1 0 1	1, 3
6	0 1 1 0	2, 3
7	0 1 1 1	1, 2, 3
8	1 0 0 0	4
9	1 0 0 1	1, 4
10	1 0 1 0	2, 4
11	1 0 1 1	1, 2, 4
12	1 1 0 0	3, 4
13	1 1 0 1	1, 3, 4
14	1 1 1 0	2, 3, 4
15	1 1 1 1	1, 2, 3, 4

Table 1: *The bit position itself tells which parity checks to apply.*

```
C1 −> 1, 3, 5, 7, 9, 11, 13, 15
C2 −> 2, 3, 6, 7, 10, 11, 14, 15
C3 −> 4, 5, 6, 7, 12, 13, 14, 15
C4 −> 8, 9, 10, 11, 12, 13, 14, 15
```

Table 2: *Each parity check affects a unique set of bit positions.*

```
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
1  0  1  0  0  1  1  0  0   1   1   0   1   0   1 xmit
               E
1  0  1  0  1  1  1  0  0   1   1   0   1   0   1 recvd
```

Table 3: *Here a transmission error occurs at bit position 5.*

```
C1 −> 1, 3,  5,  7,  9, 11, 13, 15 −> fails −> 1
C2 −> 2, 3,  6,  7, 10, 11, 14, 15 −> passes−> 0
C3 −> 4, 5,  6,  7, 12, 13, 14, 15 −> fails −> 1
C4 −> 8, 9, 10, 11, 12, 13, 14, 15 −> passes−> 0
```

Table 4: *The parity checks locate the error (0101 = 5 in binary).*

probably deduced by now that you can correct $2^m - 1$ bits of total information with m bits of parity check (subject to only 1-bit errors, of course!). So with 10 bits set aside to do the checks, you have 1023 − 10, or 1013 bits of information sent. Even if you include the additional parity bit for 2-bit errors, this represents an overhead of about 1 percent, which is not bad when you consider that you will correct all 1-bit errors at the receiver and detect blocks with 2-bit errors.

The real power of forward error correction comes from being able to do better, however. More complicated codes can correct not only single-bit errors but also multiple-bit errors. And, because for some channels the predominant mode is not single-bit errors but errors that come in bunches, there are codes that are better suited to correcting burst errors. There are also codes that handle both single-bit errors and multiple bursts. If you are interested, you can learn more by looking up the BCH and Reed Solomon codes in the references at the end of this article.

One Last Note

Communications is often thought of as getting information from here to there. Another way of looking at it is getting information from then to now—that is, all these forward error correction schemes can be applied to disk writing and reading. When you fetch a block from a disk, good disk controller software can do an FEC maneuver and fix up the block if it was written or read incorrectly.

Use of these schemes enables the cost of disk hardware to come down because lower precision mechanical assemblies can be used; systems can be designed to have higher rates of read/write errors if we know we can correct for them. Perhaps the most stunning demonstration of this phenomenon is the audio compact disc player. The digital music that is retrieved is encoded with a sophisticated Reed Solomon forward error correction code that enables magnificent sound with less than perfect media. That you can buy one of these at less than $200 is a remarkable example of intelligent hardware and software integration.

Note

1. R. W. Hamming, *Coding and Information Theory* (Englewood Cliffs, N.J.: Prentice-Hall, 1980).

Bibliography

Berlekamp, E. *Algebraic Coding Theory.* New York: McGraw Hill, 1968.

Lin, S., and Costello, D. *Error Control Coding.* Englewood Cliffs, N.J.: Prentice-Hall, 1983.

DDJ

(Listing begins on page 84.)

Vote for your favorite feature/article.
Circle Reader Service **No. 3.**

We're Ready, Are You?

68000
68010
68020

WE ARE PROUD TO ANNOUNCE THE BIRTH OF THE NEWEST MEMBERS OF OUR 68000 FAMILY . . . YOUR 68020 TOOLS ARE HERE!

TOOL KIT

- 68000/10/20 Assembler Package:
  - Macro Cross/Native Assembler
  - Linker and Librarian
  - Cross Reference Facility
  - Symbol Formatter Utility
  - Object Module Translator
- Green Hills C 68000/10/20 Optimizing Compilers
- Symbolic Debuggers

FEATURES

- Written in C; fast, accurate, portable.
- Supports 68000 and 68010.
- 5,000 line test suite included.
- EXORmacs compatible.
- Produces full listings and maps.
- Outputs S-records and Tek-Hex formats.
- Runs native or cross. Extensive libraries.
- Supports OASYS compilers.
- Generates PROMable output and PIC.
- Full Floating Point support.

AVAILABILITY

VAX, microVAX, 8600, Sun, Pyramid, Masscomp, IBM/PC, OASYS Attached Processors for VAX and PC, others. Runs under VMS, Bsd 4.2, System V, MS/DOS, dozens more.

You name it . . .

We provide a "One-Stop Shopping" service for more than 100 products running on, and/or targeting to, the most popular 32-, 16- and 8-bit micros and operating systems.

A DIVISION OF XEL Oasys

60 Aberdeen Avenue, Cambridge, MA 02138 **(617) 491-4180**

We Specialize In:
Cross/Native Compilers C, Pascal, Fortran, Cobol, Basic, APL, PL/1, Prolog, Lisp, ADA — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Translators Converters — Profilers — QA Tools — Design Tools — Comm. Tools OS Kernels — Editors — Spreadsheets — Data Bases — VAX & PC Attached Processors and more

We Support:
680xx, 80x86, 320xx, 68xx, 80xx, dozens more

AMERICAN MICRO TECHNOLOGY
presents

STARRING

XT-PLUS AMT 286 and Introducing AT *jr*

XT-PLUS

IBM PC XT Compatible, 4.77MHz Clock, 640K Mother Board, 8088 Intel Chip, Keyboard, 135 watt Power Supply, Floppy Disk Controller, Printer Port, Serial Port, Game Port, Clock w/Battery Back-up, Two Disk Drives . . . . . **$699.**

Specials *

Floppy Disc Controller . . . . . **$29.**
Monochrome Graphics Card/PP . . . . . **79.**
Disk I/O Card FDC, PP/SP, Game, Battery Clock . . . . . **79.**
1200 Baud Modem (½ size) . . **159**
XT-Mother Board "O"K expandable to 640K . . . . . **99.**
Mother Board for AT . . . . . **650.**
Keyboard AT . . . . . **65.**
20MB Drive with Controller . . **425.**
Power Supply 135 watts . . . . **65.**
Power Supply 200 watts . . . **129.**
XT Chassis . . . . . **31.**
AT Chassis . . . . . **89.**
8MHz IBM PC XT Compatible Mother Board "O"K . . . . . **175.**
30MB Drive with Controller . . **590.**
EGA Card . . . . . **275.**

AMT 286

IBM PC AT Compatible Computer (STM Board) 6 MHz, 640K Memory, Keyboard, Clock & Battery on Board, Floppy & Hard Disk Controller, 1.2MB Floppy, 192 watts Power Supply **$1499.***

AMT 286-e

IBM PC AT Compatible Computer (Atronics Mother Board), 8 MHz Clock (enhanced version), 640KB Memory expandable to 1MB, AT layout Keyboard, Floppy & Hard Disk Controller by Western Digital, 192 watts Power Supply, 1.2MB Floppy, 20MB Hard Disk, Socket for 80287 . . . . . **$1999.**

"O" Wait State Option . . . . **$300.**

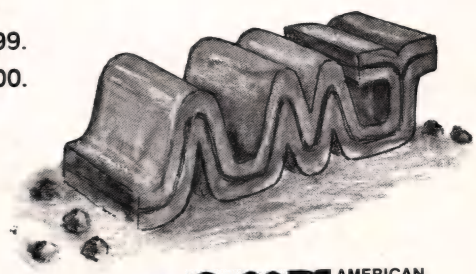
PLEASE DON'T CALL TICKETRON . . . FOR FAST SERVICE CALL US

(714) 972-2945
TWX 5106003265
(213) 477-6320 inside Los Angeles

AT *jr*

IBM PC Compatible, 2 to 5 times faster than IBM PC, Zero Wait State, Eight Slots, 135 watts Power Supply, 8 MHz & 4.77 MHz Clock (Dual Speed Hardware & Software switchable), 640KB on Mother Board, V-20 or 8088-2 Processor, AT layout Keyboard, Floppy Disk Controller, Two Drives 360KB each, Runs Lotus 123, Wordstar, dBase II & III, Flight Simulator and more **$699.***

*Prices for quantity purchases only
Registered trademark of IBM Corporation



AMT AMERICAN MICRO TECHNOLOGY
1322 E. EDINGER SANTA ANA, CA 92705

Prices and Availability subject to change without notice.

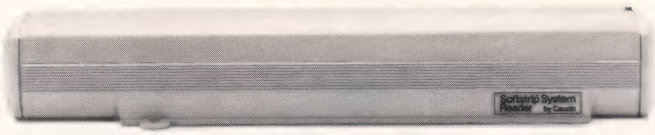
Circle **no. 297** on reader service card.

It's amazing what you can reveal when you strip.

Introducing a shape that's about to turn on an entire industry.

The Softstrip™ data strip. From Cauzin.

This new technology allows text, graphics, and data to be encoded on a strip of paper, then easily entered into your computer using a scanning device called the Cauzin Softstrip™ System Reader.



The Cauzin Softstrip System Reader replaces tedious typing by scanning the strip and reading it into your computer.

Creating a simple, reliable and cost efficient way to distribute and retrieve information.

Softstrip data strips, like those you see here, can contain anything that can be put on magnetic disks.

Facts. Figures. Software programs.

Video games. Product demonstrations. Sheet music.

The Cauzin Softstrip System Reader is now compatible with the IBM PC, Apple II and Macintosh.

A single strip can hold up to 5500 bytes of encoded data.

It can stand up to wrinkles, scratches, ink marks, even coffee stains.

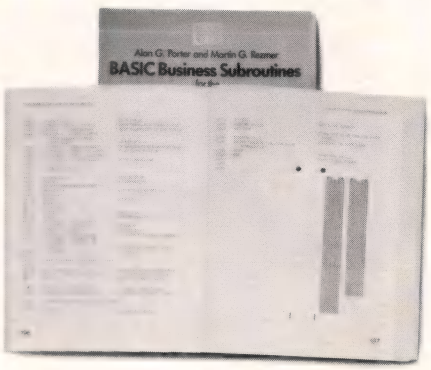
And it can be entered into your computer with a higher degree of reliability than most magnetic media.

Simply by plugging the Cauzin Reader into your serial or cassette port and placing it over the strip.

The reader scans the strip, converts it to computer code, and feeds it into any standard communication interface.

Because strips are so easy to generate, most of your favorite magazines and books will soon be using them in addition to long lists of program code.

And you'll be able to enter programs without typing a single line.



Soon everyone will be stripping as data strips appear in popular magazines, computer books and text books.

There is also software for you to generate your own strips. Letting you send everything from correspondence to business information using our new technology.

Find out how much you can reveal by stripping. Just take this ad to your computer dealer for a demonstration of the Cauzin Softstrip System Reader.

Or for more information and the name of the dealer nearest you, call Cauzin at 1-800-533-7323. In Connecticut, call 573-0150.

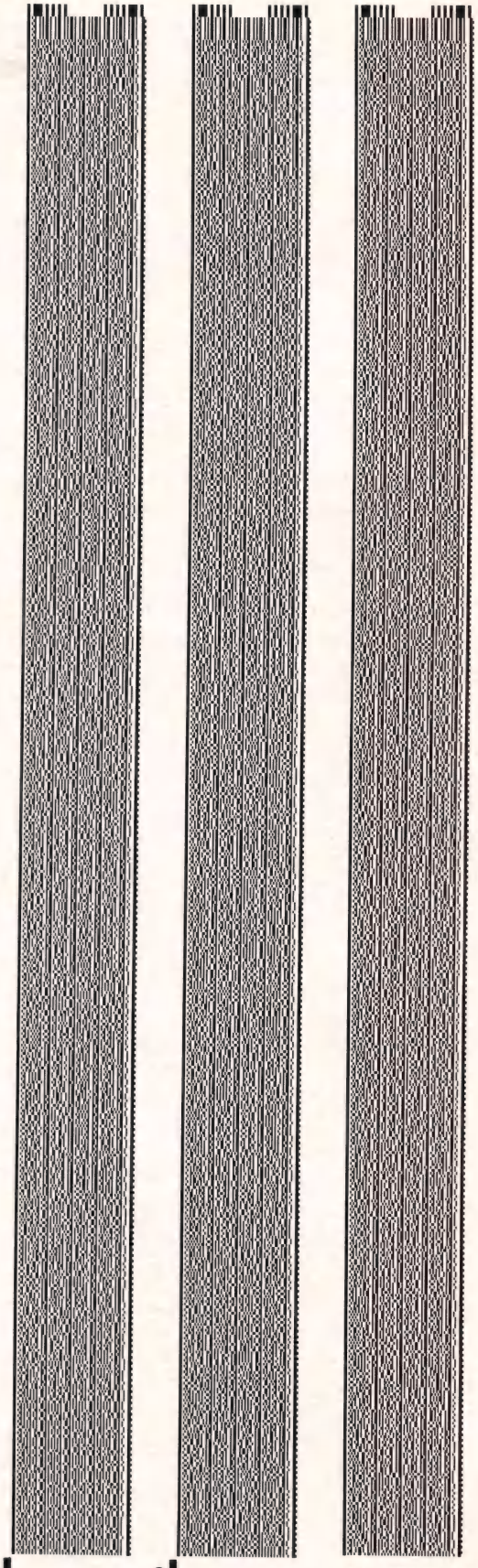
Softstrip™
COMPUTER READABLE PRINT

Cauzin Systems, Inc.
835 South Main St., Waterbury, CT 06706

Apple® and Macintosh® are registered trademarks of Apple Computer Inc., Apple® is a registered trademark of Apple Records, Inc., Softstrip® and the Softstrip® System Reader are trademarks of Cauzin Systems, Inc., IBM® is a registered trademark of IBM, Inc.

Circle no. 226 on reader service card.

MAKING THE GOOD LIFE EVEN BETTER



Someone once said that there is nothing new under the sun. Wouldn't life be boring if that were indeed true? The data strips on the right contain the program described in the article "The Game of Life in Expert-2", by Jack Park, which appears in this issue. It's a prime example of how something, in this case the game of **LIFE** itself, can, indeed be improved.

The game of **LIFE** was invented years ago by John Horton Conway. Over the years, the game has evolved into a popular cerebral exercise for programmers and math majors alike. At first the game was played on graph paper, but the advent of modern technology moved it to the computer which plays the game thousands of times faster. Now millions of computer enthusiasts are captivated by this devilishly simple, yet marvelously complex quintessential computer diversion.

The rules of the game are quite simple. Imagine that you have an infinite grid of squares, each one being either alive (on) or dead (off). Each square (called a "cell") lives or dies into the next cycle (called a "generation") based on its current state and that of its neighbors. The grid of cells is represented by a graphic display on your computer screen. After setting up an initial configuration of living and dead cells, you start the simulation. The patterns will change on the screen as cells live and die.

Mr. Park's improvement on the theme is interesting because of his approach. Instead of writing a traditional program for the simulation, he has created an array of intelligent cells using an inference engine written in Expert-2, a superset of FORTH.

Read in the data strips, following the directions that came with your Cauzin reader. You'll need the Expert-2 programming environment to operate this program. Refer to Mr. Park's article in this issue for operating instructions.

Reprinted with permission of Dr. Dobb's Journal.

StripWare Library No. 202

1 2 3

Softstrip

The CompuServe B Protocol: A Better Way to Send Files

by Levi Thomas and Nick Turner

With the B protocol, unlike the others, you let the mainframe do all the work.

Talking to the Big Boys

If you've tried to use a commercial telecommunications package to upload or download files to or from a mainframe computer, you may have experienced a few difficulties. The most popular file-transfer protocols all have problems dealing with the "big boys"—XMODEM, for example, has a tendency to "time out" when the host system experiences a momentary delay in transmission. Other problems also get in the way. Many protocols require a file length that is an exact multiple of some block size. We've run into these problems here at *DDJ* while uploading listings for the DDJ Forum on CompuServe.

In 1980, a programmer at CompuServe wrote one of the first programs that tried to fix some of these problems, calling it the CompuServe A protocol. It had numerous glitches and design problems. The B protocol, designed about the same time as Ward Christensen designed XMODEM, is the next generation. For a while CompuServe afficionados wanted to keep both the A and B protocols proprietary, but CompuServe B is now in the public domain, supported by CompuServe. Because it was designed with the idea of communicating between micros and mainframes over packet-switching networks, it incorporates several improvements that largely eliminate the problems of the other protocols. With the B protocol, unlike the others, you let the mainframe do all the work.

With the B protocol, the host (usually a mainframe) activates the protocol in your micro automatically, as soon as you have invoked the protocol through your commands to the host. You must have a terminal program running in your local system that recognizes the host's initial B protocol query and automatically invokes its own "slave" program to accept or supply the data. The host also has the ability to interrogate your micro to find out what features your program supports. B protocol supports error-corrected file transfer between computers, chiefly text file transfer between microcomputers and mainframes and binary file transfer between microcomputers with possible intermediate storage on mainframes. B protocol can transfer files of arbitrary size and supports character mapping on text transfers.

The program we describe in this article is a dumb-terminal emulator with just enough intelligence to recognize when the host is about to transfer a file up or down. It responds to the host's queries and implements the protocol transparently to the terminal user. BP.C (Listing One, page 90), the vanilla version described and supplied here in C source form, is not machine specific and should be installable in most existing terminal programs, provided you have access to the source code or information on how to connect new device drivers. We've also included some machine-language and C modules for the interface routines that will work on most MS-DOS machines.

How It Works

To transfer a file using the B protocol, first you invoke it in the host system (usually the mainframe you're calling) by sending the proper commands to it manually through your terminal interface. On CompuServe this would mean selecting the proper choice from the menu. Then the host system sends the ASCII *ENQ* character, to which your terminal program responds with *DLE 0* (data link Escape followed by an ASCII *0*). The host is acting as the master in this exchange (even though the protocol was invoked by you) because it is invoking the slave process in your micro.

Next, the host usually sends the sequence *ESC I* (Escape followed by ASCII *I*). Upon receipt, the microcomputer terminal software should transmit an identification string to the requesting computer. The identification string consists of the pound sign (#), followed by a three-letter, alphanumeric, product name code (in this case the code is *DTE*, meaning a general terminal device), a version number in decimal, a comma (,), and a comma-separated list of feature codes. The feature code list details the features supported by the particular terminal program. The codes that denote a B protocol driver are *PB* (which refers to protocol B) and *DT* (disk transfer). Additional feature codes describe other capabilities, in-

cluding terminal type and graphics support.

To start the transfer, the host sends a *DLE* followed by ASCII *B*. At this point, the main terminal loop should call *Transfer_File* (a BP.C routine) to complete the protocol sequence. *Transfer_File* returns a Boolean value indicating the success/failure of the transfer.

Routines You Provide

In order to use the program supplied here, you must provide some routines for your specific hardware. You'll need four routines to control your modem or serial port. These are used by BP.C to open, read, write, and close your modem port. You'll need a couple of routines to deal with your local keyboard and screen. The program will call them to read a character from your keyboard, to write a character to your screen, and to find out if you have decided to abort the program (for example, by pressing the Esc key). You'll need to supply two timer routines: one that sets a time out of a certain number of seconds and another that tests whether the time is up. If you don't have a hardware timer, you can simulate the effect simply by decrementing a count every time the time-out routine is called. Finally, you'll need a set of file-manipulation routines whose format will be largely dependent on exactly what operating system you're running under and what library routines you're using.

Modem Interface Routines

Open_Modem—is required to initialize the modem port to support 8-bit data without auto XON/XOFF recognition before BP.C is called. No parity checking should be done. BP.C does not depend on baud rate and stop-bit settings. **Note:** Depending on the particular machine, data may be lost when the modem port is reprogrammed for no auto XON/XOFF recognition. This data loss usually affects only the first block, which the slave software can request to be retransmited by sending a NAK.

Write_Modem—is called by BP.C. Its argument is an 8-bit character to be transmitted.

Read_Modem—is called by BP.C and returns an integer containing an 8-bit character or −1. The latter is used to indicate that no character has been received over the modem port.

Close_Modem—is required to shut down the modem port outside BP.C and to restore the machine to its original state. Certain changes made in *Open_Modem*, such as reprogramming interrupt vectors, must be undone before you exit from the program.

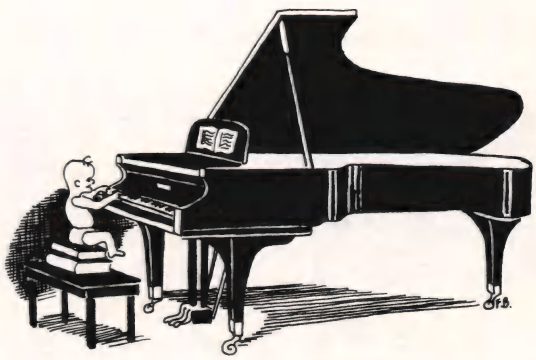
User Input/Output Routines

Read_Keyboard (from KEYBOARD .ASM, Listing Two, page 99)—returns an integer containing either ASCII-key codes, function-key codes, or −1 for no key pressed.

Wants_To_Abort (from DTE.C, Listing Three, page 100)—is called by BP.C to detect if the user has requested abort. The routine returns a Boolean *true* if abort has been indicated. Once the *Wants_To_Abort* status has been read, the status is reset. You may notice some delay before the abort request is acknowledged. The delay is because of the need to synchron-

C–terp

The C Interpreter You Won't Outgrow



C–terp will grow with you as you progress from novice through professional to guru. Unbelievable, but true, the easiest-to-use C interpreter will provide you with the most advanced programming features for upward growth. Our exclusive **object module support** enables you to add libraries (like HALO, PANEL, Windows for C, etc., or your own homebrew libraries) to C–terp as you add them to your computing repertoire. Use C–terp as a microscope on your libraries! Flip a bit and allow our **software paging** (NEW) to handle those big jobs! There are no fixed-size tables to overflow, and C–terp can be configured for different screens and screen adapters (NEW). With multiple modules and **full K&R support,** we offer a dream C environment.

- Our new improved **configurable editor** competes with anything going.
- Speed -- Linking and semi-compilation are breathtakingly fast.
- Convenience -- Errors direct you back to the editor with the cursor set to the trouble spot.
- Symbolic Debugging -- Set breakpoints, single-step, and directly execute C expressions.
- Compatibility guaranteed – batch file to link in your compiler's entire library. Supported compilers include: **Computer Innovations C86, Lattice C, Microsoft C 3.0, Mark Williams C86,** and **Aztec C.**
- Many more features including batch mode and 8087 support.

What Our Users/ Reviewers Are Saying

"... easy to use, powerful, and a timesaver."

"... we absolutely LOVE C–terp."

"... has restored my faith in interpreters."

"... a programmer's dream."

"... wonderful technical assistance."

"... increased our productivity by a factor of 40."

"... the best C product ever, in any category."

- **Price: $300.00 (Demo $45.00) MC, VISA**

Prices include documentation and shipping within U.S. PA residents add 6% sales tax. Specify compiler.

- C–terp runs on the IBM PC (or any BIOS compatible machine) under DOS 2.x and up with a suggested minimum of 256 Kb of memory. It can use all the memory available.

• C–terp is a trademark of Gimpel Software.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426

(215) 584-4261

ize the abort packet with the current protocol packet.

Put_Char (from SCREEN.ASM, Listing Four, page 104)—accepts an integer containing ASCII character codes. The characters are displayed to the user.

Timer Routines

These are routines from TIMER.ASM, Listing Five, next month.

Start_Timer—is passed an integer argument of the number of seconds to begin counting down.

Timer_Expired—returns Boolean *true* if the number of seconds set with *Start_Timer* has elapsed. If you don't have a real-time clock or timer, *Start_Timer* should set a delay counter that *Timer_Expired* decrements each time it is called, so that the number of calls to *Timer_Expired* controls time out. *Timer_Expired* is called frequently during time-out detection.

Delay (from DELAY.C, Listing Six, next month)—is called with an integer argument for the number of milliseconds to wait before returning to the calling routine. The only purpose of *Delay* is to support a wait acknowledgment request from the other computer. The wait acknowledgment can be used to request delay time needed during intensive processing.

File Input/Output Routines

These are routines from FILEIO.ASM, Listing Seven, next month. These file primitives are used to create, read, and write both text and binary files. The data should be transferred unmodified (except in character-mapping mode). The file-access mode should allow access to each byte in the file, often called "binary" or "raw" mode. The actual arguments of each of the file input/output routines will need to be those supported by the particular library you are using.

Create_File—attempts to create a file with the name supplied and returns a negative-result code (for errors) or a file handle.

Open_File—opens the file, returning a negative error code or the file handle.

Read_File—reads the specified number of bytes from the open file into the specified buffer, returning a negative error code or the number of bytes actually read.

Write_File—writes the specified number of bytes from the open file into the specified buffer, returning a negative error code or the number of bytes actually written.

Close_File—closes the file, returning a negative error code or 0.

The B protocol is not perfect, but for this application, it's clearly an improvement over the previous micro-to-micro protocols. Currently, only CompuServe supports the B protocol. The master program will eventually be available in the public domain as is the slave program presented here.

DDJ

(Listings begin on page 90.)

Vote for your favorite feature/article.
Circle Reader Service **No. 4.**

C Programmers: If you do business application development, you need

***FAST* PROGRAMMING™**

The C programming language tool for business applications.

Why Fast Programming?

Because it is a **complete** development system. You will no longer need to integrate incompatible libraries and fragments of programs.

Because finished, ready-to-compile C programs are generated.

Because there are **no run time royalties.** Not even for the utilities.

Because the resulting system is truly multiuser.

Fast Programming is $995 for a site license. VISA, MC, AMEX accepted. Currently available versions include PC-DOS, XENIX and UNIX. Call for specific machine availability.

Subject, Wills & Company
800 Enterprise Drive
Oak Brook, Illinois 60521
(312) 789-0240

The components of Fast Programming are:

C ROUTINES
Decimal math, keyin and display with windowing, extended string handling, time/date conversions, time/date edits, misc edits, sequential I/O, key sequential I/O, random I/O, indexed I/O, printer output, error handler, system interface routines and more.

C PROGRAM GENERATOR
Maintenance and data entry program generator, report program generator, processing program generator, I/O routines generator. The generators provide many user exits in the generated source code. This allows for custom coding without changing the generated C source program. This means that no custom coding is lost when programs are regenerated.

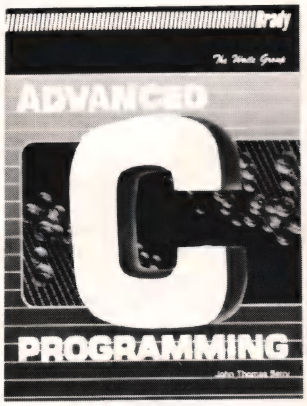
RUN TIME UTILITIES
SORT, INDEX, REFORMAT, BUILD, CREATE, DUMP, CHAIN, ENCODE, FILECHK, LIST, RENAME, COPY and DELETE.

MENU SYSTEM

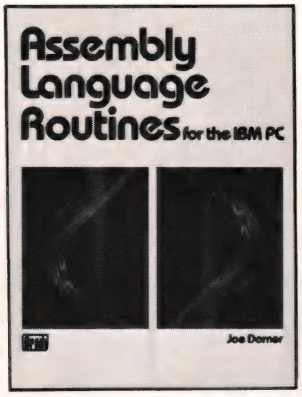
Circle **no. 288** on reader service card.

BRADY Knows Programming.

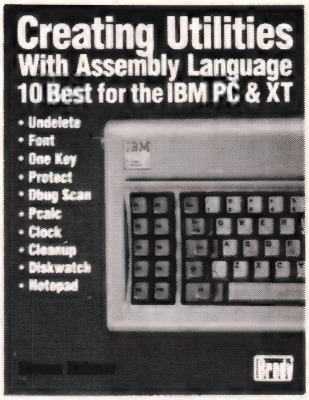
You'll learn to whip every element of your programming into shape with the latest information and guidance by America's foremost technical experts. Just call toll-free or use the coupon below to order today.



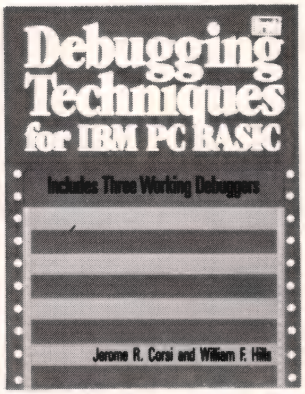
1. Beyond the basics, this guide explores new structured concepts to analyze and solve problems in C with tools of modularity, input and output functions, arrays and structures, and bit manipulation. $21.95



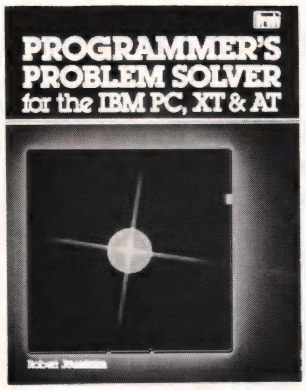
2. You learn by example with this guide through hundreds of subroutine listings. Discover how to enhance high level language programs (esp. BASIC) to quickly manipulate and sort large data files, generate graphics, integrate arrays, and use interrupts to access DOS. $17.95



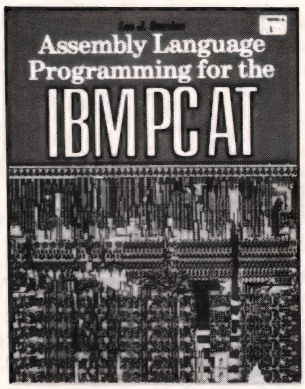
3. Learn the techniques used for creating assembly language utilities. Includes 10 of the most popular utilities such as DBUG SCAN, CLOCK, UNDELETE, ONE KEY, PCALC calculator and notepad and five others. $21.95 (Disk available)



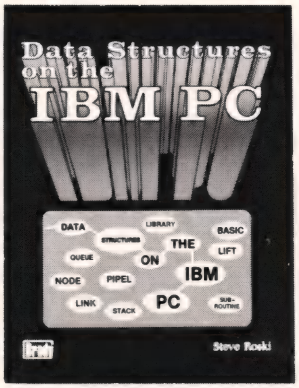
4. Includes code listings for three working debuggers including single-stepping, cross referencing, and mapping utilities. $19.95 (Disk available)



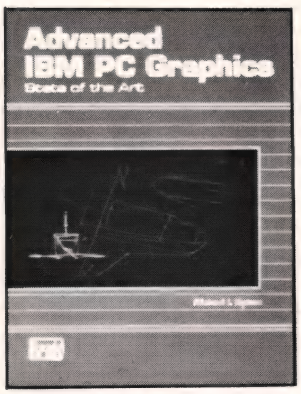
5. A definitive reference text for advanced programmers. You'll find over 150 discussions of common hardware-control tasks (in BASIC, Pascal, or C) as well as assembler overlays, drivers, and real-time operators. $22.95



6. Perfect for both beginners and experienced programmers, you'll find everything from the basics of computer numbering through to step-by-step instructions for using the IBM Macro Assembler. With complete coverage of BIOS and a library of over 30 macros for faster programming. $21.95 (Disk available)



7. Here's a compendium of many of the most useful, but often neglected, advanced programming concepts. A tutorial in format that uses BASIC for examples, it covers techniques such as: linked data structures; recursion; pipelining; and dynamic storage allocation. Includes listings for 25 subroutines. $21.95 (Disk available)



8. The title might say "advanced" but you'll find a guide that begins with the fundamentals of BASIC graphics and takes you through truly sophisticated 3-D assembly routines. Includes block graphics, creating a graphics editor, directly programming IBM's color graphics adapter, and much more. $21.95

Now at your book or computer store.
Or order toll-free today: **800-624-0023** In New Jersey: 800-624-0024

BRADY COMMUNICATIONS COMPANY, INC.
c/o Prentice Hall
P.O. Box 512, W. Nyack, NY 10994

Circle the numbers of the titles you want below. (Payment must be enclosed; or, use your charge card.) Add $1.50 for postage and handling.
Enclosed is check for $_____ or charge to
☐ MasterCard ☐ VISA

Acc't # _____ Exp. date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____
(New Jersey residents, please add applicable sales tax.)
Dept. 3

GR-BKKD-BW(9)

1 (0-89303-473-8) 2 (0-89303-409-6) 3 (0-89303-584-X) 4 (0-89303-587-4)
5 (0-89303-787-7) 6 (0-89303-484-3) 7 (0-89303-481-9) 8 (0-89303-476-2)

Brady

Circle **no. 285** on reader service card.

The SwyftCard: Jef Raskin's New User Interface

by Dave Caulkins

The word processor is the heart of the system.

SwyftCard
Information Appliance Inc., 1014 Hamilton Ct., Menlo Park, CA 94025, (415) 328-5160; $89.95

The SwyftCard from Information Appliance is a new personal computer environment for the Apple IIe and IIc. It provides a word processor from which you can easily perform other tasks, such as disk file access, calculations via BASIC, printing, and communications with a modem. The environment is noteworthy because it is philosophically at odds with the popular interface made of icons, windows, and mouse that is used on the Macintosh and other computer systems. Significantly, SwyftCard's designer is Jef Raskin, one of the initiators of the Macintosh project at Apple Computer.

I tested SwyftCard on an Apple IIe, but as this issue goes to press, Information Appliance has announced a version for the IIc also. To use SwyftCard on a IIe, you need an 80-column card, a video monitor suitable for 80-column display, and a disk drive. The SwyftCard supports a wide variety of printers; several require no setup, and many can be installed by using the Calc command.

SwyftCard consists of a three-chip board that plugs into slot 3, a set of stick-on labels for several keys, a tutorial disk, and an excellent manual. The tutorial will help you get started with SwyftCard by guiding you through a series of short lessons on the use of its features.

The word processor is the heart of the SwyftCard system. It has been carefully designed to achieve several goals:

- Speed: For text processing and floppy-disk access, SwyftCard is significantly faster than more expensive systems. All the text is in RAM, so disk access time doesn't slow SwyftCard down. Half the system is implemented in tokenized Forth and half in assembly language, resulting in fast operation. Each SwyftCard file occupies a single floppy disk and can expand to 40,000 characters, or roughly 14 pages of single-spaced text. Disk operations take less than seven seconds in all cases.
- Simplicity: There are only ten basic commands: Insert, Delete, Print, Leap, Creep, Page, Calc, Print, Send, and Disk. The last three are for input/output and are not used as frequently as the others. Most commands are implemented with one or two keystrokes using a few dual-purpose, specially labeled, SwyftCard function keys. After a little practice, users learn the commands, and text processing becomes fast and easy.
- Optional environment: You can load an ordinary Apple disk, and the operating system works normally, as SwyftCard hides behind the scenes until you need it.

Dave Caulkins, 437 Mundel Way, Los Altos, CA 94022

How SwyftCard Works

Leaping

SwyftCard allows you to move through text using two Leap keys: the open-apple and closed-apple keys on either side of the space bar. To make learning easy, a set of stick-on labels is provided to indicate which keys are used for special SwyftCard functions. Leaping means moving from one place in the text to another immediately.

Suppose the phrase *The number is less than the numerator* appears between the cursor and the end of the file and you want to locate the cursor on the *n* in *numerator*. Press the right Leap key as you type "n−u−m−e" to create a search pattern. After you have typed the "n," the cursor leaps to the next instance of *n* in the file, after the *u*, to the next instance of *nu*, and so on until it stops on the *n* in the word *numerator*.

If this isn't the instance of the word you want, press the Leap key, and the Leap Again key (Tab), to find the next occurrence of the letters. Leap Again auto-repeats; if you hold it down for more than half a second, the cursor will move rapidly from one instance of the pattern to the next. If the cursor arrives at the end of the file, it wraps back to the beginning. It will continue its forward direction until the Leap keys are released.

If you wish to leap to a point between the cursor and the beginning of the file, use the same procedure

with the left Leap key, and the search will take place in the reverse direction.

Lowercase letters in a leap pattern match both uppercase and lowercase letters in the text. Uppercase letters in the pattern match only uppercase in the text. You can leap from word to word (press Leap and the space bar), paragraph to paragraph (press Leap and Return), or page to page (press Leap and the Page key [Esc]).

Creeping

Moving the cursor over just a few characters is called creeping in the SwyftCard manual. To creep, press and release the right Leap key, and the cursor moves a character to the right. To creep in reverse, press the left Leap key.

Deleting

Deletes can also be done to the left or right of the cursor. While you are typing, if you press the Del key, the character to the left is erased as with the backspace key on a typewriter. After you leap or creep to a new place in the file, characters to the right of the cursor are deleted. The appearance of the cursor itself indicates which delete is operative. A narrow cursor is one character wide and appears when right delete is in effect; the wide cursor is two characters across with the cursor and the reverse-video character split.

Highlighting

Another useful SwyftCard feature is the highlighting of text. Highlighted text can be deleted, saved in a buffer for later insertion, printed, or telecommunicated. Highlighting takes place by pressing and releasing a Leap key, moving the cursor, and then pressing both Leap keys simultaneously. Any amount of text can be highlighted, from two characters to the entire document. Pressing Delete will remove all highlighted text from the screen and place it in a buffer. To restore the text, locate the cursor where you want the text and press Insert (Control-A). The deleted text remains in the buffer until you highlight and delete other text.

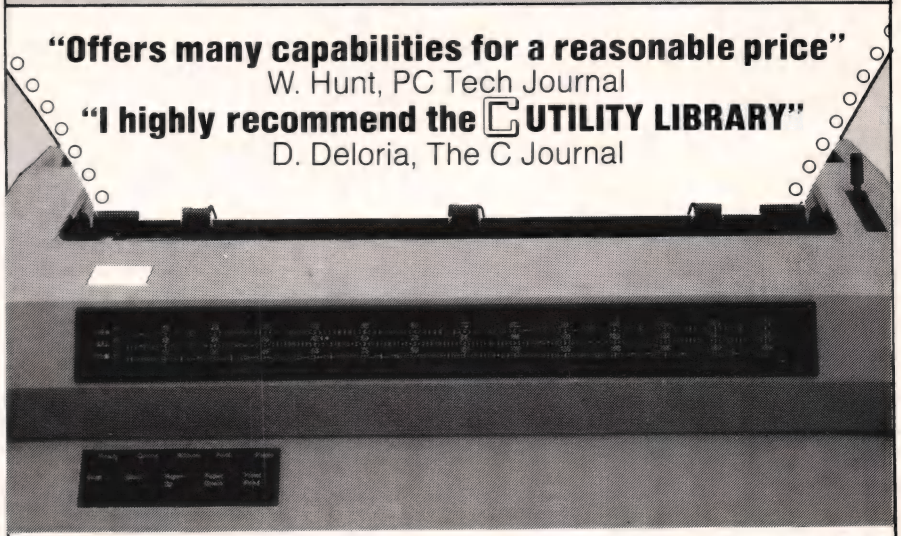
The Disk Command

You use the Disk key to read from or write a file to the disk. SwyftCard's method of handling disk files automatically takes care of whether reading or writing is required. Let's say you want to save some text. Put a floppy disk in the drive, and press the Disk key. SwyftCard notes if the disk is empty and that text is in RAM and deduces that a disk write is needed. When text has been saved, the cursor blinks rapidly to indicate that RAM and disk contents are identical.

On the other hand, if you are in the middle of writing and try to load a new file from a different floppy, SwyftCard will observe that some changes in RAM have not been saved and refuse to overwrite. A beep serves to remind you that the disk for the old file should be inserted in the drive to save changes. If you really do not want to save your changes, the procedure is to delete the text. These are examples of a well-planned user interface, designed to save the user from accidental, catastrophic errors, which all computer users have experienced at some time. Another example of this care is a SwyftCard com-

C Programmer Essentials

"Offers many capabilities for a reasonable price"
W. Hunt, PC Tech Journal
"I highly recommend the C UTILITY LIBRARY"
D. Deloria, The C Journal



C ESSENTIALS **$100**
200 functions: video, strings, keyboard, directories, files, time/date and more. Source code is 95% C. Comprehensive manual with plenty of examples. Demo programs on diskette. Upgrade to THE C UTILITY LIBRARY for $95.

THE C UTILITY LIBRARY **$185**
Thousands in use world wide. 300 functions for serious software developers. The C ESSENTIALS plus "pop-up" windows, business graphics, data entry, DOS command and program execution, polled async communications, sound and more.

ESSENTIAL GRAPHICS **$250**
Fast, powerful, and easy to use. Draw a pie or bar chart with one function. Animation (GET and PUT), filling (PAINT) and user definable patterns. IBM color, IBM EGA and Hercules supported (more soon). NO ROYALTIES. Save $50 when purchased with above libraries. Available February, 1986.

Compatible with Microsoft Ver. 3, Lattice, Aztec, Mark Williams, CI-C86, DeSmet, and Wizard C Compilers. IBM PC/XT/AT and true compatibles.

C Compiler Packages: Microsoft C - 319, Lattice or CI-C86 compilers -$329. Save $40 - $50 when purchasing compiler and library combinations. Specify C compiler and version number when ordering. Add $4 for UPS or $7 for UPS 2-day. NJ residents add 6% sales tax. Visa, MC, Checks, PO's.

ESI ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, NJ 07040 914/762-6605

Circle no. 138 on reader service card.

SWYFTCARD
(continued from page 43)

mand that will make a disk look blank, in effect destroying anything on it and allowing you to write to it. Because this command can affect your data and is thus dangerous, it does not use the Disk key and is difficult to execute accidentally.

The Calc Command

The Calc command causes a BASIC statement to be executed in the file; for example, if you type "?34 + 78" and then highlight it and press the Calc key, the answer *112* will appear in place of the BASIC statement in your text. More complex executions are also possible. If you type

```
10 FOR I = 1 TO 31
20 PRINT "JANUARY"; I
30 NEXT I
RUN
```

then highlight this program and press the Calc key, the following calendar for January will be placed in your file where the program was:

```
JANUARY 1
JANUARY 2
JANUARY 3
    .
    .
JANUARY 30
JANUARY 31
```

Almost all BASIC commands will work, but the size of the BASIC program is limited to 900 bytes in the compacted internal form. Some uses of BASIC are dangerous—*CALL*, *PEEK*, and *POKE* can zap your file if used incorrectly or with values that interfere with the SwyftCard.

Awkward Moments

When the SwyftCard has used all available RAM memory, it will beep each time you press a key. To create room, execute at least two deletions. One is insufficient because the delete buffer continues to take up the same amount of RAM. At this point it is clearly advisable to save the file, insert a new disk, delete part of the text, and continue. This is one of the few awkward operations of SwyftCard. An improvement would be some sort of warning when all but 50 or 100 bytes of RAM had been used to allow for a more graceful conclusion of an editing session.

The SwyftCard approach of allowing each disk to hold one file is acceptable for a machine such as the Apple II, but as memory and hard disks continue to drop in price, the SwyftCard environment will have to adapt to machines with greater internal and external storage capabilities. These implementations will require a file management system and some scheme for mapping files into a range of RAM memory sizes.

These minor complaints are far less important than the many impressive characteristics of SwyftCard, including speed, ease of use, and diversity. Overall, SwyftCard offers a strikingly innovative user interface that deserves the attention of users and software developers interested in advancing the cause of usable computers.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 5.**

"Revelation is 50% harder to learn than dBase III™, and 5 times more powerful."

John Dunbar
Dunbar and Company
Application Designers

That's a small price to pay for power. Especially when it helps a programmer like John Dunbar create LAN databases for the Houston Rockets, Compaq Computer Corporation and Houston Natural Gas.

Revelation has all the tools Dunbar needs including a program generator to slash development time and a robust language that puts C to shame.

Prove it to yourself. The Revelation Demo/ Tutorial is only $24.95. Technical specifications are free. Send today for more information.

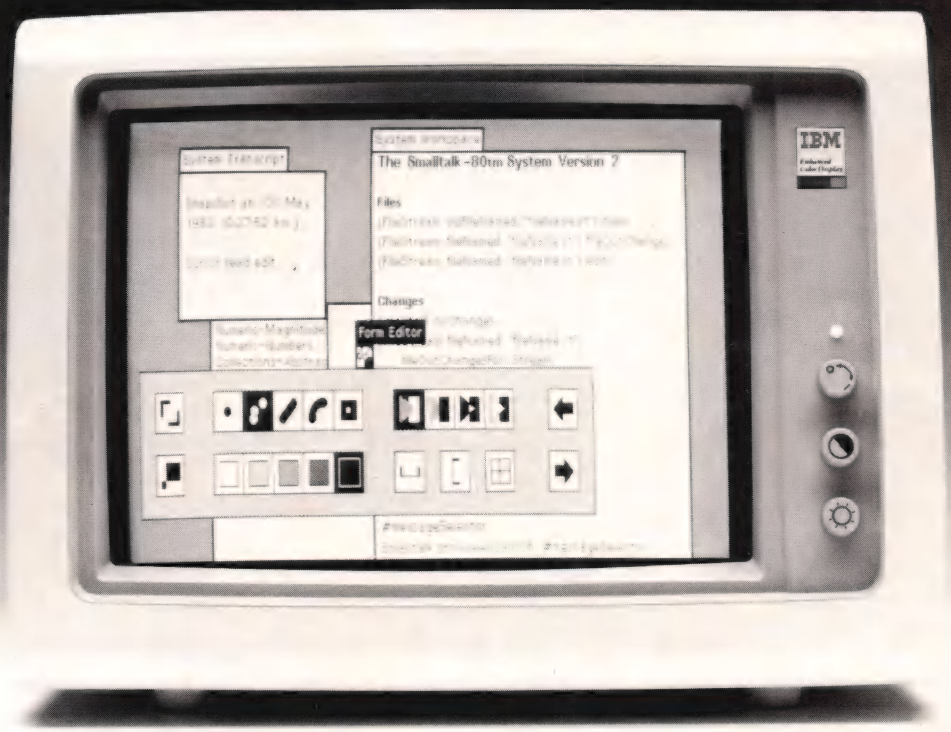
COSMOS™

Cosmos, Inc., 19530 Pacific Highway South, Suite 102
Seattle, WA 98188, (206) 824-9942, Telex: 9103808627
dBase III is a trademark of Ashton-Tate.

Circle **no. 282** on reader service card.

$995 COMPLETE

Real Smalltalk-80.™ Real performance. Real affordable.



Softsmarts introduces Smalltalk for the IBM PC/AT.

It's real because it's Smalltalk-80. Only Smalltalk-80 offers the complete set of Smalltalk classes, system browsers, workspaces, text and graphics editors, symbolic debuggers, and inspectors—all the tools you need for object-oriented programming, refined by more than 10 years of Xerox research.

It's fast because it's optimized to work on the IBM PC/AT with bit-mapped graphics, a mouse, and a window interface. So, you can forget everything you ever heard about slow Smalltalk.

It's affordable because it runs on an IBM Personal Computer. It's the only Smalltalk-80 that does. In fact, it's the most affordable way to run Smalltalk-80, Version 2. And it's available today.

Smalltalk-80 for the IBM PC/AT. From Softsmarts.
4 Skyline Drive, Woodside, CA. 94062
415-327-8100.

softsmarts, inc.
intelligent software

Smalltalk-80 is a trademark of Xerox Corporation. IBM is a trademark of International Business Machines Corporation.

Circle **no. 180** on reader service card.

Listing Seventeen (continued from May)

```
IMPLEMENTATION MODULE CodeGenerator;
(* Uses information supplied by Parser, OperationCodes, *)
(* and SyntaxAnalyzer to produce the object code.       *)

   FROM Strings IMPORT
      Length, CompareStr;

   FROM SymbolTable IMPORT
      FillSymTab, ReadSymTab;

   FROM Parser IMPORT
      TOKEN, OPERAND, OpLoc, SrcLoc, DestLoc;

   FROM LongNumbers IMPORT
      LONG, LongAdd, LongSub, LongInc, LongDec,
      LongClear, CardToLong, LongToCard, LongToInt,
      LongCompare, AddrBoundW, AddrBoundL;

   FROM OperationCodes IMPORT
      ModeTypeA, ModeTypeB, ModeA, ModeB, Instructions;

   FROM ErrorX68 IMPORT
      ErrorType, Error;

   FROM SyntaxAnalyzer IMPORT
      OpMode, Xtype, SizeType, OpConfig, Src, Dest,
      Size, Op, AddrModeA, AddrModeB, InstSize,
      GetValue, GetSize, GetInstModeSize, GetOperand, GetMultReg;


   CONST
      JMP = {14, 11, 10, 9, 7, 6};
      JSR = {14, 11, 10, 9, 7};
      RTE = {14, 11, 10, 9, 6, 5, 4, 1, 0};
      RTR = {14, 11, 10, 9, 6, 5, 4, 2, 1, 0};
      RTS = {14, 11, 10, 9, 6, 5, 4, 2, 0};
      TRAPV = {14, 11, 10, 9, 6, 5, 4, 2, 1};
      STOP = {14, 11, 10, 9, 6, 5, 4, 1};
      LINK = {14, 11, 10, 9, 6, 4};
      SWAP = {14, 11, 6};
      UNLK = {14, 11, 10, 9, 6, 4, 3};
      Quote = 47C;


   VAR
   (*---
      (* Defined in DEFINITION MODULE *)
      LZero, AddrCnt : LONG;
      Pass2 : BOOLEAN;
                                        ---*)
      AddrAdv : LONG;
      TempL : LONG;      (* Temporary variables *)
      TempI : INTEGER;
      TempC : CARDINAL;
      BrValue : LONG;    (* Used to calculate relative branches *)
      RevBr : BOOLEAN;


   PROCEDURE BuildSymTable (VAR AddrCnt : LONG;
                            Label, OpCode : TOKEN; SrcOp, DestOp : OPERAND);
   (* Builds symbol table from symbolic information of Source File *)

      VAR
         Value : LONG;
         Full : BOOLEAN;
         PseudoOp : BOOLEAN;

      BEGIN
         Value := LZero;
         AddrAdv := LZero;
         InstSize := 0;
         PseudoOp := FALSE;
         Size := S0;

         IF Length (OpCode) = 0 THEN
            RETURN;    (* Nothing added to symbol table, AddrCnt not changed *)
         END;

         GetSize (OpCode, Size);

         IF CompareStr (OpCode, "ORG") = 0 THEN
            GetValue (SrcOp, AddrCnt);
            AddrBoundW (AddrCnt);
            Value := AddrCnt;
            PseudoOp := TRUE;
         ELSIF CompareStr (OpCode, "EQU") = 0 THEN
            GetValue (SrcOp, Value);
            PseudoOp := TRUE;
         ELSIF CompareStr (OpCode, "DC") = 0 THEN
            CASE Size OF
               Word  :  AddrBoundW (AddrCnt);
            |  Long  :  AddrBoundL (AddrCnt);
            |  Byte  :  ;
            END;

            IF SrcOp[0] = Quote THEN    (* String Constant *)
               TempC := Length (SrcOp);
               IF TempC > 2 THEN
                  InstSize := TempC - 2;
               END;
            ELSE
               InstSize := ORD (Size);
            END;
            CardToLong (InstSize, AddrAdv);
            Value := AddrCnt;
            PseudoOp := TRUE;
         ELSIF CompareStr (OpCode, "DS") = 0 THEN
            GetValue (SrcOp, AddrAdv);
            Value := AddrCnt;
            PseudoOp := TRUE;
         ELSIF CompareStr (OpCode, "EVEN") = 0 THEN
            AddrBoundW (AddrCnt);
            Value := AddrCnt;
            PseudoOp := TRUE;
         ELSIF CompareStr (OpCode, "END") = 0 THEN
            PseudoOp := TRUE;
         ELSE
            Value := AddrCnt;
         END;

         IF Length (Label) # 0 THEN
            FillSymTab (Label, Value, Full);
            IF Full THEN
               Error (0, SymFull);
            END;
         END;

         IF NOT PseudoOp THEN

            Instructions (OpCode, OpLoc, Op, AddrModeA, AddrModeB);

            AddrBoundW (AddrCnt);
            Src.Loc := SrcLoc;   Dest.Loc := DestLoc;
            GetOperand (SrcOp, Src);
            GetOperand (DestOp, Dest);
            InstSize := 2;    (* minimum size of instruction *)

            IF Brnch IN AddrModeA THEN
               IF Size # Byte THEN
                  INC (InstSize, 2);
               END;
            ELSIF DecBr IN AddrModeA THEN
               INC (InstSize, 2);
            ELSE
               IF (Op = JMP) OR (Op = JSR) THEN    (* Allows for 'JMP.S' *)
                  IF (Size = Byte) AND (Src.Mode = AbsL) THEN
                     Src.Mode := AbsW;
                  END;
               END;

               TempC := GetInstModeSize (Src.Mode, Size, InstSize);
               TempC := GetInstModeSize (Dest.Mode, Size, InstSize);
            END;

            IF (Src.Mode = Imm) AND
              ((Data911 IN AddrModeA) OR (Data03 IN AddrModeA) OR
              (Data07 IN AddrModeA) OR (CntR911 IN AddrModeA)) THEN
               (* Quick instruction *)
               InstSize := 2;
            END;
            CardToLong (InstSize, AddrAdv);
         END;
      END BuildSymTable;


   PROCEDURE MergeModes (VAR SrcOp, DestOp : OPERAND;
                         VAR ObjOp, ObjSrc, ObjDest : LONG;
                         VAR nO,    nS,     nD      : CARDINAL);
   (*  Uses information from Instructions & GetOperand (among others)  *)
   (*  to complete calculation of Object Code.                          *)
   (*  Op, AddrModeA, AddrModeB, Size, and Src & Dest records are all   *)
   (*  Global variables imported from the SyntaxAnalyzer MODULE.        *)

      CONST
         (* BITSETs of the modes MISSING from effective address modes  *)
         aa = {};                  (* Effective addressing - all modes *)
         dea = {1};                (* Data effective addressing        *)
         mea = {1, 0};             (* Memory effective addressing      *)
         cea = {11, 4, 3, 1, 0};   (* Control effective addressing     *)
         aea = {11, 10, 9};        (* Alterable effective addressing   *)
         xxx = {15, 14, 13};       (* extra modes: CCR/SR/USP          *)
         (* 2 "AND" masks to turn off switch bits for shift/rotate *)
         Off910 = {15, 14, 13, 12, 11, 8, 7, 6, 5, 4, 3, 2, 1, 0};
         Off34 = {15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 2, 1, 0};

      VAR
         M : CARDINAL;
         i : CARDINAL;
         Ext : BITSET;       (* Bit pattern for instruction extension word *)
         ExtL : LONG;
         Xext : BITSET;
         Quick : BOOLEAN;


      PROCEDURE OperExt (VAR EA : OpConfig);
      (* Calculate Operand Extension word, and check range of Operands *)

         VAR
            GoodCard, GoodInt : BOOLEAN;

         BEGIN
            GoodCard := LongToCard (EA.Value, TempC);
            GoodInt := LongToInt (EA.Value, TempI);

            CASE EA.Mode OF
               AbsL     :  ;   (* No range checking needed *)
            |  AbsW     :  IF NOT GoodCard THEN
                              Error (EA.Loc, SizeErr);
                           END;
            |  ARDisp,
               PCDisp   :  IF NOT GoodInt THEN
                              Error (EA.Loc, SizeErr);
                           END;
            |  ARDisX,
               PCDisX   :  IF (TempI < -128) OR (TempI > 127) THEN
                              Error (EA.Loc, SizeErr);
                           END;
                           Xext := BITSET (EA.Xn * 4096);
                           IF EA.X = Areg THEN
                              Xext := Xext + {15};
                           END;
                           IF EA.Xsize = Long THEN
                              Xext := Xext + {11};
                           END;
                           CardToLong (CARDINAL (Xext), TempL);
                           EA.Value[3] := TempL[3];
                           EA.Value[4] := TempL[4];
            |  Imm      :  IF Size = Long THEN
                              (* No range check needed *)
                           ELSE
                              IF GoodInt THEN
                                 IF Size = Byte THEN
                                    IF (TempI < -128) OR (TempI > 127) THEN
                                       Error (EA.Loc, SizeErr);
                                    END;
                                 END;
                              ELSE
                                 Error (EA.Loc, SizeErr);
                              END;
                           END;
            ELSE
               (* No Action *)
            END;
         END OperExt;

      PROCEDURE EffAdr (VAR EA : OpConfig; Bad : BITSET);
      (* adds effective address field to Op (BITSET representing opcode) *)

         VAR
            M : CARDINAL;
            i : CARDINAL;
            Xext : BITSET;

         BEGIN
            M := ORD (EA.Mode);

            IF M IN Bad THEN
```

```
            Error (EA.Loc, ModeErr);
            RETURN;
         ELSIF M > 11 THEN
            RETURN;
         ELSIF M < 7 THEN
            Op := Op + BITSET (M * 8) + BITSET (EA.Rn);
         ELSE   (*    7  <=  M  <=  11    *)
            Op := Op + (5, 4, 3) + BITSET (M - 7);
         END;

         OperExt (EA);
      END EffAdr;

   BEGIN   (* MergeModes *)
      ExtL := LZero;
      Quick := FALSE;

      (* Check for 5 special cases first *)

      IF (Op = RTE) OR (Op = RTR) OR (Op = RTS) OR (Op = TRAPV) THEN
         IF Src.Mode # Null THEN
            Error (SrcLoc, OperErr);
         END;
      END;

      IF Op = STOP THEN
         IF (Src.Mode # Imm) OR (Dest.Mode # Null) THEN
            Error (SrcLoc, OperErr);
         END;
      END;

      IF Op = LINK THEN
         Op := Op + BITSET (Src.Rn);
         IF (Src.Mode # ARDir) OR (Dest.Mode # Imm) THEN
            Error (SrcLoc, ModeErr);
         END;
      END;

      IF Op = SWAP THEN
         IF EA05f IN AddrModeB THEN
            (* Ignore, this is PEA instruction! *)
         ELSE
            Op := Op + BITSET (Src.Rn);
            IF (Src.Mode # DReg) OR (Dest.Mode # Null) THEN
               Error (SrcLoc, OperErr);
            END;
         END;
      END;

      IF Op = UNLK THEN
         Op := Op + BITSET (Src.Rn);
         IF (Src.Mode # ARDir) OR (Dest.Mode # Null) THEN
            Error (SrcLoc, OperErr);
         END;
      END;

      (* Now do generalized address modes *)

      IF (Ry02 IN AddrModeA) AND (Rx911 IN AddrModeA) THEN
         Op := Op + BITSET (Src.Rn) + BITSET (Dest.Rn * 512);
         (* Now do some error checking! *)
         IF RegMem3 IN AddrModeA THEN
            IF Src.Mode = DReg THEN
               IF Dest.Mode # DReg THEN
                  Error (DestLoc, ModeErr);
               END;
            ELSIF Src.Mode = ARPre THEN
               Op := Op + {3};
               IF Dest.Mode # ARPre THEN
                  Error (DestLoc, ModeErr);
               END;
            ELSE
               Error (SrcLoc, OperErr);
            END;
         ELSE
            IF Src.Mode = ARPost THEN
               IF Dest.Mode # ARPost THEN
                  Error (DestLoc, ModeErr);
               END;
            ELSE
               Error (SrcLoc, OperErr);
            END;
         END;
      END;

      IF Data911 IN AddrModeA THEN
         Quick := TRUE;
         IF Src.Mode = Imm THEN
            IF LongToInt (Src.Value, TempI)
             AND (TempI > 0)
              AND (TempI <= 8) THEN
               IF TempI < 8 THEN   (* Data of 8 is coded as 000 *)
                  Op := Op + BITSET (TempI * 512);
               END;
            ELSE
               Error (SrcLoc, SizeErr);
            END;
         ELSE
            Error (SrcLoc, OperErr);
         END;
      END;

      IF CntR911 IN AddrModeA THEN
         (* Only Shift/Rotate use this *)
         IF Dest.Mode = DReg THEN
            Op := (Op * Off910) + BITSET (Dest.Rn);
            CASE Size OF
               Byte : ;
            |  Word : Op := Op + {6};
            |  Long : Op := Op + {7};
            END;
            IF Src.Mode = DReg THEN
               Op := Op + {5} + BITSET (Src.Rn * 512);
            ELSIF Src.Mode = Imm THEN
               Quick := TRUE;
               (* Range Check *)
               IF LongToInt (Src.Value, TempI)
                AND (TempI > 0)
                 AND (TempI <= 8) THEN
                  IF TempI < 8 THEN   (* Data of 8 is coded as 000 *)
                     Op := Op + BITSET (TempI * 512);
                  END;
               ELSE
                  Error (SrcLoc, SizeErr);
               END;
            ELSE
               Error (SrcLoc, OperErr);
            END;
         ELSIF Dest.Mode = Null THEN
```

(continued on next page)

SOFTWARE DESIGN ENGINEERS

Because of our rapid growth, we're looking for nothing less than the top Software Design Engineers in the country — wizards who can combine microcomputer software development skills with creative imagination. You will work in networking, sophisticated graphics, operating systems, compilers, productivity software, product marketing and other exciting areas. In each project, you'll have the opportunity to explore totally new realms of microcomputer software.

The right candidates will have a degree in Computer Science, or a related field with a minimum of 3 years relevant work experience in the microcomputer industry. Experience in micros, systems programming using "C", Pascal, or Assembler, operating systems and working in small teams is essential to these positions. Experience in 8086 Assembler, XENIX/UNIX, MS-DOS, and microprocessors (286, 8086, 68000) is desired.

Microsoft is located in the Seattle area of the beautiful Pacific Northwest, where the amenities of civilization live side by side with the grandeur of mountains and pine forests. It's one of the most prized living environments in the United States.

Come to Microsoft, where you will have the space and flexibility to prove to the computer industry that you're the best at what you do. Microsoft offers excellent opportunities and a complete benefits package. Send your resume to: **MICROSOFT CORPORATION, Human Resources, Dept. SS, 16011 NE 36th Way, Box 97017, Redmond, WA 98073-9717.** An Equal Opportunity Employer.

MICROSOFT®

The High Performance Software

Listing Seventeen (listing continued)

```
      Op := (Op * Off34) + {7, 6};
      EffAdr (Src, (mea + aea));
   ELSE
      Error (SrcLoc, OperErr);
   END;
END;

IF Data03 IN AddrModeA THEN
   Quick := TRUE;
   IF Src.Mode = Imm THEN
      IF LongToInt (Src.Value, TempI)
       AND (TempI >= 0)
        AND (TempI < 16) THEN
         Op := Op + BITSET (TempI);
      ELSE
         Error (SrcLoc, SizeErr);
      END;
   ELSE
      Error (SrcLoc, OperErr);
   END;
END;

IF Data07 IN AddrModeA THEN
   Quick := TRUE;
   IF (Src.Mode = Imm) AND (Dest.Mode = DReg) THEN
      IF LongToInt (Src.Value, TempI)
       AND (TempI >= -128)
        AND (TempI <= 127) THEN
         Op := Op + (BITSET (TempI) * {7, 6, 5, 4, 3, 2, 1, 0})
                  + BITSET (Dest.Rn * 512);
      ELSE
         Error (SrcLoc, SizeErr);
      END;
   ELSE
      Error (SrcLoc, OperErr);
   END;
END;

IF OpM68D IN AddrModeA THEN
   IF Dest.Mode = DReg THEN
      Op := Op + BITSET (Dest.Rn * 512);
      IF (Src.Mode = ARDir) AND (Size = Byte) THEN
         Error (SrcLoc, SizeErr);
      END;
   ELSE   (* Assume Src.Mode = DReg -- Error trapped elsewhere *)
      Op := Op + BITSET (Src.Rn * 512);
      Op := Op + {8};
   END;

   CASE Size OF
      Byte : ;
   |  Word : Op := Op + {6};
   |  Long : Op := Op + {7};
   END;
END;

IF OpM68A IN AddrModeA THEN
   IF Dest.Mode = ARDir THEN
      Op := Op + BITSET (Dest.Rn * 512);
   ELSE
      Error (DestLoc, ModeErr);
   END;

   CASE Size OF
      Byte : Error (OpLoc, SizeErr);
   |  Word : Op := Op + {7, 6};
   |  Long : Op := Op + {8, 7, 6};
   END;
END;

IF OpM68C IN AddrModeA THEN
   IF Dest.Mode = DReg THEN
      Op := Op + BITSET (Dest.Rn * 512);
   ELSE
      Error (DestLoc, ModeErr);
   END;

   CASE Size OF
      Byte : IF Src.Mode = ARDir THEN
                Error (OpLoc, SizeErr);
             END;
   |  Word : Op := Op + {6};
   |  Long : Op := Op + {7};
   END;
END;

IF OpM68X IN AddrModeA THEN
   IF Src.Mode = DReg THEN
      Op := Op + BITSET (Src.Rn * 512);
   ELSE
      Error (SrcLoc, ModeErr);
   END;

   CASE Size OF
      Byte : Op := Op + {8};
   |  Word : Op := Op + {8, 6};
   |  Long : Op := Op + {8, 7};
   END;
END;

IF OpM68S IN AddrModeA THEN
   IF Src.Mode = DReg THEN
      Op := Op + BITSET (Src.Rn);
   ELSE
      Error (SrcLoc, ModeErr);
   END;

   CASE Size OF
      Byte : Error (OpLoc, SizeErr);
   |  Word : Op := Op + {7};
   |  Long :   Op := Op + {7, 6};
   END;
END;

IF OpM68R IN AddrModeA THEN
   IF (Src.Mode = DReg) AND (Dest.Mode = ARDisp) THEN
      CASE Size OF
         Byte : Error (OpLoc, SizeErr);
      |  Word : Op := Op + {8, 7};
      |  Long : Op := Op + {8, 7, 6};
      END;
      Op := Op + BITSET (Src.Rn * 512) + BITSET (Dest.Rn);
   ELSIF (Src.Mode = ARDisp) AND (Dest.Mode = DReg) THEN
      CASE Size OF
         Byte : Error (OpLoc, SizeErr);
      |  Word : Op := Op + {8};
      |  Long : Op := Op + {8, 6};
      END;
      Op := Op + BITSET (Src.Rn) + BITSET (Dest.Rn * 512);
   ELSE
      Error (SrcLoc, ModeErr);
   END;
END;

IF OpM37 IN AddrModeA THEN
   IF (Src.Mode = DReg) AND (Dest.Mode = DReg) THEN
      Op := Op + {6} + BITSET (Src.Rn * 512) + BITSET (Dest.Rn);
   ELSIF (Src.Mode = ARDir) AND (Dest.Mode = ARDir) THEN
      Op := Op + {6, 3} + BITSET (Src.Rn * 512) + BITSET (Dest.Rn);
   ELSIF (Src.Mode = ARDir) AND (Dest.Mode = DReg) THEN
      Op := Op + {7, 3} + BITSET (Dest.Rn * 512) + BITSET (Src.Rn);
   ELSIF (Src.Mode = DReg) AND (Dest.Mode = ARDir) THEN
      Op := Op + {7, 3} + BITSET (Src.Rn * 512) + BITSET (Dest.Rn);
   ELSE
      Error (SrcLoc, ModeErr);
   END;
END;

IF Bit811 IN AddrModeB THEN
   IF Src.Mode = DReg THEN
      Op := Op + {8} + BITSET (Src.Rn * 512);
   ELSIF Src.Mode = Imm THEN
      Op := Op + {11};
   ELSE
      Error (SrcLoc, ModeErr);
   END;
END;

IF Size67 IN AddrModeB THEN
   CASE Size OF
      Byte : ;(* No action -- bits already 0's *)
   |  Word : Op := Op + {6};
   |  Long : Op := Op + {7};
   END;
END;

IF Size6 IN AddrModeB THEN
   CASE Size OF
      Byte : Error (OpLoc, SizeErr);
   |  Word : (* No Action -- BIT is already 0 *)
   |  Long : Op := Op + {6};
   END;
END;

IF Size1213A IN AddrModeB THEN
   CASE Size OF
      Byte : Op := Op + {12};
   |  Word : Op := Op + {13, 12};
   |  Long : Op := Op + {13};
   END;
END;

IF Size1213 IN AddrModeB THEN
   Op := Op + BITSET (Dest.Rn * 512);
   CASE Size OF
      Byte : Error (OpLoc, SizeErr);
   |  Word : Op := Op + {13, 12};
   |  Long : Op := Op + {13};
   END;
END;

IF EA05a IN AddrModeB THEN
   IF (Dest.Mode = DReg) OR (Dest.Mode = ARDir) THEN
      EffAdr (Src, ea);
   ELSE
      Error (DestLoc, ModeErr);
   END;
END;

IF EA05b IN AddrModeB THEN
   IF Dest.Mode = DReg THEN
      EffAdr (Src, dea);
      Op := Op + BITSET (Dest.Rn * 512);
   ELSE
      Error (DestLoc, ModeErr);
   END;
END;

IF EA05c IN AddrModeB THEN
   EffAdr (Dest, {11, 1});
END;

IF EA05d IN AddrModeB THEN
   EffAdr (Dest, aea);
   IF (Dest.Mode = ARDir) AND (Size = Byte) THEN
      Error (OpLoc, SizeErr);
   END;
END;

IF EA05e IN AddrModeB THEN
   IF Dest.Mode = Null THEN
      EffAdr (Src, (dea + aea));
   ELSIF (Src.Mode = Imm) OR (Src.Mode = DReg) THEN
      EffAdr (Dest, (dea + aea));
   ELSE
      Error (SrcLoc, ModeErr);
   END;
END;

IF EA05f IN AddrModeB THEN    (* LEA & PEA / JMP & JSR *)
   EffAdr (Src, cea);
   IF Rx911 IN AddrModeA THEN
      IF Dest.Mode = ARDir THEN
         Op := Op + BITSET (Dest.Rn * 512);
      ELSE
         Error (DestLoc, ModeErr);
      END;
   ELSE
      IF Dest.Mode # Null THEN
         Error (DestLoc, OperErr);
      END;
   END;
END;

IF EA05x IN AddrModeB THEN
   IF Dest.Mode = DReg THEN
      EffAdr (Src, dea);
   ELSIF Src.Mode = DReg THEN
      EffAdr (Dest, mea + aea);
   ELSE
      Error (SrcLoc, OperErr);
   END;
END;

IF EA05y IN AddrModeB THEN
   IF Dest.Mode = DReg THEN
      EffAdr (Src, ea);
      IF (Src.Mode = ARDir) AND (Size = Byte) THEN
```

(continued on page 50)

WIZARD C

The new **Wizard C version 3.0** sets new records for all out speed! It leaves other C compilers in the dust! When your project depends on that last ounce of speed, choose Wizard.

The following SIEVE benchmark was run without register variable declarations on an IBM/PC with 640K memory and an 8087.

	Exec Time	Code Size	EXE Size
Wizard C 3.0	**: 6.8**	**130**	**7,766**
Microsoft	:11.5	186	7,018
Lattice	:11.8	164	20,068

Fast executable code, with multiple levels of optimization.

Six memory Models, supporting up to 1 megabytye of code and data, plus mixed model programming.

Effective error diagnosis, including multiple source file cross-checking of function calls.

A comprehensive runtime library, including fully portable C functions, MSDOS and 8086 functions.

8086, 8087, 80186 and 80286 hardware support.

Full Library Source Code included with package.

ROM based application support, including a ROM development package available to create Intel Hex files.

Fully ANSI compatible language features.

"...The compiler's performance makes it very useful in serious software development."

PC Tech Journal
January, 1986

" Wizard's got the highest marks for support."

"The Wizard Compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal
August, 1985

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

For MSDOS applications, we provide the most features of any C compiler, plus a full range of third party software:

- PANEL
- Greenleaf Libraries
- Essential Software Library
- PLINK86
- Pfix86plus
- Microsoft Assembler 3.0

For stand-alone applications, we supply a ROM development package that carries your program all the way to Intel Hex files ready for a PROM burner.

For debugging, the compiler emits full Intel debugging information including local symbol and type information.

Wizard C	**$450.00**
ROM Development Package	**$350.00**
Combined Package	**$750.00**

(617) 641-2379

WIZARD SYSTEMS SOFTWARE, INC.

11 Willow Court, Arlington, MA 02174

MasterCard VISA AMERICAN EXPRESS

Listing Seventeen (listing continued)

```
            Error (OpLoc, SizeErr);
         END;
      ELSIF Src.Mode = DReg THEN
         EffAdr (Dest, (mea + aea));
      ELSE
         Error (SrcLoc, ModeErr);
      END;
   END;

   IF EA05z IN AddrModeB THEN
      IF Src.Mode = MultiM THEN
         EffAdr (Dest, (mea + aea + {3}));
         GetMultReg (SrcOp, (Dest.Mode = ARPre), SrcLoc, Ext);
      ELSIF Dest.Mode = MultiM THEN
         EffAdr (Src, (mea + {11, 4}));
         GetMultReg (DestOp, (Src.Mode = ARPre), DestLoc, Ext);
         Op := Op + {10};   (* set direction *)
      ELSE
         Error (SrcLoc, OperErr);
      END;

      INC (nO, 4);   (* extension is part of OpCode *)
      INC (InstSize, 2);
      CardToLong (CARDINAL (Ext), ExtL);
   END;

   IF EA611 IN AddrModeB THEN
      IF Dest.Mode = CCR THEN
         Op := {14, 10, 7, 6};
         EffAdr (Src, dea);
      ELSIF Dest.Mode = SR THEN
         Op := {14, 10, 9, 7, 6};
         EffAdr (Src, dea);
      ELSIF Src.Mode = SR THEN
         Op := {14, 7, 6};
         EffAdr (Dest, dea + aea);
      ELSIF Dest.Mode = USP THEN
         Op := {14, 11, 10, 9, 6, 5};
         IF Src.Mode = ARDir THEN
            Op := Op + BITSET (Src.Rn);
         ELSE
            Error (SrcLoc, ModeErr);
         END;
      ELSIF Src.Mode = USP THEN
         Op := {14, 11, 10, 9, 6, 5, 3};
         IF Dest.Mode = ARDir THEN
            Op := Op + BITSET (Dest.Rn);
         ELSE
            Error (DestLoc, ModeErr);
         END;
      ELSE
         EffAdr (Src, (ea + xxx));
         IF (Size = Byte) AND (Src.Mode = ARDir) THEN
            Error (SrcLoc, SizeErr);
         END;

         M := ORD (Dest.Mode);
         IF (M IN (dea + aea)) OR (M > 11) THEN
            Error (DestLoc, ModeErr);
         ELSIF M < 7 THEN
            Op := Op + BITSET (M * 64) + BITSET (Dest.Rn * 512);
         ELSE   (* 7 <= M <= 11 *)
            Op := Op + {8, 7, 6} + BITSET ((M - 7) * 512);
         END;

         OperExt (Dest);
      END;
   END;

   IF (Dest.Mode = CCR) AND (Src.Mode = Imm) THEN
      IF (Size67 IN AddrModeB)
       AND (EA05e IN AddrModeB)
        AND (Exten IN AddrModeB) THEN
         IF 10 IN Op THEN   (* NOT ANDI/EORI/ORI *)
            Error (DestLoc, ModeErr);
         ELSE
            Op := Op * {15, 14, 13, 12, 11, 10, 9, 8};   (* AND mask *)
            Op := Op + {5, 4, 3, 2};                     (*  OR mask *)
         END;
      END;
   END;

   IF (Dest.Mode = SR) AND (Src.Mode = Imm) THEN
      IF (Size67 IN AddrModeB)
       AND (EA05e IN AddrModeB)
        AND (Exten IN AddrModeB) THEN
         IF 10 IN Op THEN   (* NOT ANDI/EORI/ORI *)
            Error (DestLoc, ModeErr);
         ELSE
            Op := Op * {15, 14, 13, 12, 11, 10, 9, 8};   (* AND mask *)
            Op := Op + {6, 5, 4, 3, 2};                  (*  OR mask *)
         END;
      END;
   END;

   CardToLong (CARDINAL (Op), ObjOp);
   INC (InstSize, 2);
   INC (nO, 4);
   IF nO > 4 THEN
      FOR i := 1 TO 4 DO   (* move ObjOp -- make room for extension *)
         ObjOp[i + 4] := ObjOp[i];
         ObjOp[i] := ExtL[i];
      END;
   END;

   nS := GetInstModeSize (Src.Mode, Size, InstSize);
   ObjSrc := Src.Value;
   nD := GetInstModeSize (Dest.Mode, Size, InstSize);
   ObjDest := Dest.Value;

   IF Quick THEN
      InstSize := 2;
      nS := 0;   nD := 0;
   END;
   CardToLong (InstSize, AddrAdv);

END MergeModes;


TYPE
   DirType = (None, Org, Equ, DC, DS, Even, End);

PROCEDURE ObjDir (OpCode : TOKEN; SrcOp : OPERAND; Size : SizeType;
                  VAR AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
                  VAR   nA,      nO,     nS,      nD     : CARDINAL) : DirType;
(* Generates Object Code for Assembler Directives *)

   VAR
      Dir : DirType;
      i, j : CARDINAL;
      LongString : ARRAY [1..20] OF INTEGER;

   BEGIN
      AddrAdv := LZero;

      IF CompareStr (OpCode, "ORG") = 0 THEN
         GetValue (SrcOp, AddrCnt);
         AddrBoundW (AddrCnt);
         Dir := Org;
      ELSIF CompareStr (OpCode, "EQU") = 0 THEN
         GetValue (SrcOp, ObjSrc);
         nS := 8;
         Dir := Equ;
      ELSIF CompareStr (OpCode, "DC") = 0 THEN
         CASE Size OF
            Word  :  AddrBoundW (AddrCnt);
          | Long  :  AddrBoundL (AddrCnt);
          | Byte  :  ;
         END;

         IF SrcOp[0] = Quote THEN   (* String constant *)
            TempC := Length (SrcOp);
            IF TempC > 2 THEN
               InstSize := TempC - 2;   (* Don't count the Quotes *)
            END;

            i := 1;   j := 20;
            WHILE i <= InstSize DO   (* Change from ASCII to LONG *)
               CardToLong (ORD (SrcOp[i]), TempL);
               LongString[j] := TempL[2];
               LongString[j - 1] := TempL[1];
               INC (i);   DEC (j, 2);
            END;

            i := 1;   INC (j);
            WHILE j <= 20 DO   (* Left Justify String *)
               LongString[i] := LongString[j];
               INC (i);   INC (j);
            END;

            DEC (i);
            WHILE i > 16 DO   (* Transfer 2 bytes to OpCode *)
               ObjOp[i - 16] := LongString[i];
               INC (nO);   DEC (i);
            END;

            WHILE i > 8 DO   (* Transfer 4 bytes to Source Operand *)
               ObjSrc[i - 8] := LongString[i];
               INC (nS);   DEC (i);
            END;

            WHILE i > 0 DO   (* Transfer 4 bytes to Destination Operand *)
               ObjDest[i] := LongString[i];
               INC (nD);   DEC (i);
            END;

            IF SrcOp[InstSize + 1] # Quote THEN
               Error ((SrcLoc + InstSize + 1), OperErr);
            END;
         ELSE   (* not a string constant *)
            GetValue (SrcOp, ObjSrc);
            InstSize := ORD (Size);
            nS := InstSize * 2;
         END;
         CardToLong (InstSize, AddrAdv);
         nA := 6;
         Dir := DC;
      ELSIF CompareStr (OpCode, "DS") = 0 THEN
         GetValue (SrcOp, AddrAdv);
         nA := 6;   nS := 2;   ObjSrc := LZero;
         Dir := DS;
      ELSIF CompareStr (OpCode, "EVEN") = 0 THEN
         AddrBoundW (AddrCnt);
         Dir := Even;
      ELSIF CompareStr (OpCode, "END") = 0 THEN
         nA := 6;
         Dir := End;
      ELSE
         Dir := None;
      END;

      RETURN (Dir);
   END ObjDir;


PROCEDURE AdvAddrCnt (VAR AddrCnt : LONG);
(* Advances the address counter based on the length of the instruction *)
   BEGIN
      LongAdd (AddrCnt, AddrAdv, AddrCnt);
   END AdvAddrCnt;


PROCEDURE GetObjectCode (Label, OpCode : TOKEN;
                         SrcOp, DestOp : OPERAND;
                         VAR AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
                         VAR   nA,      nO,     nS,      nD     : CARDINAL);
(* Determines the object code for the operation as well as the operands *)
(* Returns each (up to 3 fields), along with the length of each.       *)

   VAR
      Dummy : BOOLEAN;
      Dir : DirType;

   BEGIN
      AddrAdv := LZero;
      InstSize := 0;
      nA := 0;   nO := 0;   nS := 0;   nD := 0;

      IF Length (OpCode) = 0 THEN
         (* ensure no code generated *)
         RETURN;
      END;

      GetSize (OpCode, Size);

      Dir := ObjDir (OpCode, SrcOp, Size,
                     AddrCnt, ObjOp, ObjSrc, ObjDest,
                       nA,      nO,     nS,      nD     );

      IF (Length (Label) # 0) AND (Dir # Equ) THEN
      (* Check for phase error *)
         Dummy := ReadSymTab (Label, TempL, Dummy);
         IF LongCompare (TempL, AddrCnt) # 0 THEN
            Error (0, Phase);
         END;
```

(continued on page 52)

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS®

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

6/86 DDJ

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. Lattice is the registered trademark of Lattice, Inc. PLINK86 and PLIB86 are trademarks of Phoenix Software Associates Ltd. UNIX is the trademark of AT&T. Copyright © 1985 by SAS Institute Inc. Printed in the USA.

Listing Seventeen (listing continued)

```
        END;

        IF Dir = None THEN    (* Instruction *)
           AddrBoundW (AddrCnt);
        ELSE
           RETURN;
        END;

        Instructions (OpCode, OpLoc, Op, AddrModeA, AddrModeB);
        Src.Loc := SrcLoc;   Dest.Loc := DestLoc;
        GetOperand (SrcOp, Src);    (* Src & Dest are RECORDS *)
        GetOperand (DestOp, Dest);

        IF DecBr IN AddrModeA THEN    (* Decrement & Branch *)
           IF Src.Mode # DReg THEN
              Error (SrcLoc, ModeErr);
           END;

           BrValue := Dest.Value;
           TempL := AddrCnt;
           TempC := 32767;    (* Maximum Branch *)
           LongInc (TempL, 2);    (* move past instruction for Rel Adr Calc *)

           IF LongCompare (BrValue, TempL) < 0 THEN
              RevBr := TRUE;
              LongSub (TempL, BrValue, BrValue);
              INC (TempC);    (* can branch 1 farther in reverse *)
           ELSE
              RevBr := FALSE;
              LongSub (BrValue, TempL, BrValue);
           END;

           CardToLong (TempC, TempL);    (* Maximum Branch distance *)

           IF LongCompare (BrValue, TempL) > 0 THEN
              Error (DestLoc, BraErr);
           END;
           IF RevBr THEN    (* Make Negative *)
              LongSub (LZero, BrValue, BrValue)
           END;

           CardToLong (4, AddrAdv);
           nA := 6;    nO := 4;    nS := 4;
           CardToLong (CARDINAL (Op + BITSET (Src.Rn)), ObjOp);
           ObjSrc := BrValue;
           RETURN;
        END;

        IF Brnch IN AddrModeA THEN    (* Branch *)
           BrValue := Src.Value;    (* Destination of Branch *)
           TempL := AddrCnt;
           LongInc (TempL, 2);

           IF Size # Byte THEN    (* Byte Size ---> Short Branch *)
              TempC := 32767;    (* Set maximum branch distance *)
           ELSE
              TempC := 127;
           END;

           CASE LongCompare (BrValue, TempL) OF
              -1 :  (* Reverse Branch *)
                    RevBr := TRUE;
                    INC (TempC);    (* can branch 1 farther in reverse *)
                    LongSub (TempL, BrValue, BrValue);
           |  +1 :  (* Forward Branch *)
                    RevBr := FALSE;
                    LongSub (BrValue, TempL, BrValue);
           |   0 :  IF Size = Byte THEN
                       Error (SrcLoc, BraErr);
                    END;
           END;

           CardToLong (TempC, TempL);

           IF LongCompare (BrValue, TempL) > 0 THEN
              Error (SrcLoc, BraErr);
           END;

           IF RevBr THEN
              LongSub (LZero, BrValue, BrValue);    (* Make negative *)
           END;

           IF Size # Byte THEN
              InstSize := 4;
              nS := 4;
              ObjSrc := BrValue;
           ELSE
              InstSize := 2;
              Dummy := LongToInt (BrValue, TempI);
              Op := Op + (BITSET (TempI) * {7, 6, 5, 4, 3, 2, 1, 0});
           END;

           nA := 6;    nO := 4;
           CardToLong (InstSize, AddrAdv);
           CardToLong (CARDINAL (Op), ObjOp);
           RETURN;
        END;

        nA := 6;
        IF (Op = JMP) OR (Op = JSR) THEN    (* Allows for 'JMP.S' *)
           IF (Size = Byte) AND (Src.Mode = AbsL) THEN
              Src.Mode := AbsW;
           END;
        END;
        MergeModes (SrcOp, DestOp, ObjOp, ObjSrc, ObjDest, nO, nS, nD);

     END GetObjectCode;


BEGIN    (* MODULE Initialization *)
   LongClear (LZero);    (* Used as a constant *)
   AddrCnt := LZero;
   Pass2 := FALSE;
END CodeGenerator.
```

End Listing Seventeen

Listing Eighteen

```
IMPLEMENTATION MODULE SyntaxAnalyzer;
(* Analyzes the operands to provide information for CodeGenerator *)

   FROM Conversions IMPORT
      StrToCard;

   FROM Strings IMPORT
      Length;

   FROM LongNumbers IMPORT
      LONG, LongAdd, LongSub, CardToLong, StringToLong;

   FROM SymbolTable IMPORT
      SortSymTab, ReadSymTab;

   FROM ErrorX68 IMPORT
      ErrorType, Error;

   FROM Parser IMPORT
      OPERAND, SrcLoc;

   FROM CodeGenerator IMPORT
      LZero, AddrCnt, Pass2;    (* BOOLEAN Switch *)


   CONST
      Zero = 30H;    (* The Ordinal value of the Character '0' *)
      Seven = 37H;    (* The Ordinal value of the Character '7' *)
      Quote = 47C;

(*---
   TYPE
      OpMode = (DReg,       (* Data Register *)
                ARDir,      (* Address Register Direct *)
                ARInd,      (* Address Register Indirect *)
                ARPost,     (* Address Register with Post-Increment *)
                ARPre,      (* Address Register with Pre-Decrement *)
                ARDisp,     (* Address Register with Displacement *)
                ARDisX,     (* Address Register with Disp. & Index *)
                AbsW,       (* Absolute Word (16-bit Address) *)
                AbsL,       (* Absolute Word (32-bit Address) *)
                PCDisp,     (* Program Counter Relative, with Displacement *)
                PCDisX,     (* Program Counter Relative, with Disp. & Index *)
                Imm,        (* Immediate *)
                MultiM,     (* Multiple Register Move *)
                SR,         (* Status Register *)
                CCR,        (* Condition Code Register *)
                USP,        (* User's Stack Pointer *)
                Null);      (* Error Condition, or Operand missing *)

      Xtype = (X0, Dreg, Areg);
      SizeType = (S0, Byte, Word, S3, Long);

      OpConfig = RECORD                    (* OPERAND CONFIGURATION *)
                    Mode : OpMode;
                    Value : LONG;
                    Loc : CARDINAL;        (* Location of Operand on line *)
                    Rn : CARDINAL;         (* Register number *)
                    Xn : CARDINAL;         (* Index Reg. nbr. *)
                    Xsize : SizeType;      (* size of Index *)
                    X : Xtype;             (* Is index Data or Address reg? *)
                 END;


   VAR
      Size : SizeType;        (* size for OpCode *)
      AbsSize : SizeType;     (* size of operand (Absolute only) *)
      InstSize : CARDINAL;    (* Size of instruction, including operands *)
      AddrModeA : ModeA;      (* Addressing modes for this instruction *)
      AddrModeB : ModeB;      (*                ditto                  *)
      Op : BITSET;            (* Raw bit pattern for OpCode *)
      Src, Dest : OpConfig;
                                                                    ---*)


   PROCEDURE CalcValue (Operand : OPERAND; VAR Value : LONG);
   (* Calculates left and right values for GetValue *)

      VAR
         Full : BOOLEAN;
         Neg : BOOLEAN;
         Dup : BOOLEAN;
         Num : CARDINAL;
         NumSyms : CARDINAL;

      BEGIN
         IF Operand[0] = '-' THEN
            Neg := TRUE;
            Operand[0] := '0';
         ELSE
            Neg := FALSE;
         END;

         IF StrToCard (Operand, Num) THEN
            (* It is a number *)
            CardToLong (Num, Value);
            IF Neg THEN
               LongSub (LZero, Value, Value);
            END;
         ELSIF StringToLong (Operand, Value) THEN
            (* It is a HEX number *)
         ELSIF (Operand[0] = Quote) AND (Operand[2] = Quote) THEN
            CardToLong (ORD (Operand[1]), Value);
         ELSIF (Length (Operand) = 1) AND (Operand[0] = '*') THEN
            Value := AddrCnt;
         ELSE
            (* It is a label, but may be undefined! *)
            IF NOT Pass2 THEN
               SortSymTab (NumSyms);
            END;
            IF NOT ReadSymTab (Operand, Value, Dup) THEN
               Error (SrcLoc, Undef);
            END;
            IF Dup THEN
               Error (SrcLoc, SymDup);
            END;
         END;
      END CalcValue;


   PROCEDURE GetValue (Operand : OPERAND; VAR Value : LONG);
   (* determines value of operand (in Decimal, HEX, or via Symbol Table) *)

      VAR
         TempOp : OPERAND;
         TempVal : LONG;
         c, op : CHAR;
         i, j : CARDINAL;

         InQuotes : BOOLEAN;
```

(continued on page 54)

THE PROGRAMMER'S SHOP™

31 Day RISK-FREE TRIAL on any product in this ad.

C Programmers: 7 Ways to Increase Productivity

SORT/MERGE With RECORD SELECTION & OUTPUT REFORMATTING
with OPT-TECH SORT

New 3.0 version is even faster and more powerful. Improve your system's performance with OPT-TECH SORT. OPT-TECH includes:

- CALLable and Standalone use
- All major languages
- Variable and fixed length
- Up to 10 sort/select fields
- Autoselect of RAM or disk
- Options: dBASE, BTrieve files
- 1 to 10 files input
- No software max for # records
- Full memory utilization
- All common field types
- Bypass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds.

MSDOS $135

C DYNAMO! WINDOWING: Full C Source, No Royalties
POWER WINDOWS and C FUNCTION LIBRARY

Power Windows covers all the bases: overlays, borders, 1-2-3 style or pop-up menus/help windows, zap instantly on/off screen, status lines, horizontal/vertical scrolling, color control or highlighting, word-wrap, files to windows, keyboard to windows. Powerful, easy to use, integrated error messages, thorough documentation. Supports IBM monochrome or color.

C Function Library - includes 325 fundamental functions with readable source and thorough documentation.

No matter what you have, you need these. Best value available. Highly recommended!

Power Windows MSDOS Only $119
C Function Library MSDOS Only $119

db_VISTA
First Database Exclusively for C is also Royalty-Free

— *"If you are looking for a sophisticated C programmer's database, this is it."* — Dave Schmitt, President, Lattice, Inc.

Designed exclusively for C, db_VISTA is a royalty-free programmer's DBMS. Take full advantage of C through ease of use, portability, and efficiency. You optimize for speed and efficient disk storage.

Multiple key records, fast B-tree indexing, virtual memory disk accessing. Tailor db_VISTA to your needs by using only those features you require. Optional dBASE, R:BASE, and ASCII file transfer utilities make moving to db_VISTA a snap.

MSDOS, Unix, Xenix, MacIntosh. Single user Source $459. Object $179. Multiuser Source $929. Object $450

Add Full Modem Control to Your Program
Greenleaf Comm Library

Writing and debugging communication programs can be difficult and frustrating. Use stable, reliable and comprehensive existing code. Communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt-driven system.

Included are 3 library/object files, 220 functions, 230 page manual, **complete source code.** Library supports Microsoft 3.0, Lattice 3.0, Aztec, and others. Hayes-compatible modem commands, and a complete sample file transfer program. **PCDOS $149**

Cross Compiler with COMPLETE SOURCE: QCX

Expand your audience with a cross-compiler and get full source code to the compiler and library. QCX can help you learn techniques for compiler design and parsing while providing you with a low-cost development tool. Hosted on MSDOS, Unix (or Xenix), or Vax VMS and generates ROMable Z80 assembly code. Cross assemblers and linkers are separately available.

QCX supports long integers and single-precision floating point, with only minor restrictions to K&R standard. Arguments to functions are reversed, which allows any function to have a variable number of arguments.

AO, an optional host optimizer, is available for only $125. Buy QCX for $465 and we'll include AO FREE!

Consider the native MSDOS compiler for $125.

MSDOS $465

Even for Small Files: Convenient, Fast Access
CBTREE — Only $99

Why spend time writing file management code when you can use consistent, flexible, documented, professional functions? Even multiuser record locking and variable-length records are supported.

Add, delete, and update without needing to reindex. Store keys and record locations in B + trees.

You can access any record or group of records by the value of a user specific key. Search your files from any point, forward or backward.

Full, balanced B-tree support includes use of multiple keys, unlimited number and length of keys.

Use this powerful ISAM, even if you've previously done without.

Learn how to write systems for managing large files by using CBTREE source as a guide. Modify it and transfer it to another operating environment without royalties.

MSDOS $99

Comprehensive C Development Library
C-Worthy by Custom Design Systems

C-WORTHY eliminates the writing of routine code and frees you to work on what makes your programs unique. Includes 425 pages of documentation with an in-depth tutorial and source code to a sample program.

A complete, consistent, and interrelated set of subsystems and functions facilitates keyboard handling, background procedures, list manipulation, screen handling, menu management, windowing, error reporting, context-sensitive help, DOS interfacing, and MORE.

Now you can support incompatible machines (like IBM PC, VICTOR 9000, Texas Instruments Professional, NEC APCIII, Wang PC, and HP 150) with the same .EXE file. No recompilation or relinking is required. All machine-dependent aspects of each microcomputer are isolated in a separate runtime overlay file which is loaded into the computer's memory along with the application at runtime. C-Worthy applications can also be executed on networks running Novell's NetWare.

The C-Worthy development utilities Help Librarian, Message Librarian, and Error Librarian allow applications to support alternate languages (like French, German, etc.) with the same source code. C-Worthy's philosophy of program development is revolutionary, encouraging modular design through the use of action procedures as arguments to functions. C-Worthy's unique design approach provides application developers with a consistent and intuitive user interface with features for both novice and advanced users. No royalties. For Lattice C and others.

MSDOS $295

Dear C Programmer:

You want the best development software for your needs. These products will help you:

- Speed your development efforts
- Write even better programs
- Increase productivity
- Reduce your programming frustration

We carry over 100 C compilers, interpreters, support libraries, debuggers, and addons, specifically designed for C programmers using MSDOS or PCDOS. Call one of our knowledgeable consultants - toll free - for details, comparisons, or for one of our specially prepared packets on C.

There is no obligation. You must be completely satisfied with the products you purchase or you will receive a full refund or replacement. You risk nothing with our 31 day risk-free trial on any product in this ad.

Yours for more productive programming,

Bruce W. Lynch

Bruce W. Lynch, President

Call for a catalog, literature, advice and service you can trust

NEW HOURS
8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™
128-D Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531

3/86

"You've got everything I've heard of, and much I haven't! . . . Normally, I expect my money to be "fan letter" enough, but you people are SUPER!"
— Shel Hall
Artell Corp.

Circle **no. 143** on reader service card.

Listing Eighteen (listing continued)

```
    BEGIN
       i := 0;
       Value := LZero;
       InQuotes := FALSE;
       op := '+';
       REPEAT
          j := 0;
          LOOP
             c := Operand[i];
             TempOp[j] := c;
             IF c = Quote THEN
                InQuotes := NOT InQuotes;
             END;
             INC (i);   INC (j);
             IF c = 0C THEN
                EXIT;
             END;
             IF (c = '+') AND (NOT InQuotes) THEN
                EXIT;
             END;
             IF (c = '-') AND (i > 1) AND (NOT InQuotes) THEN
                EXIT;
             END;
          END;
          TempOp[j - 1] := 0C;   (* in case c is +/- *)
          CalcValue (TempOp, TempVal);
          IF op = '-' THEN
             LongSub (Value, TempVal, Value);
          ELSE
             LongAdd (Value, TempVal, Value);
          END;
          op := c;
       UNTIL op = 0C;
    END GetValue;


PROCEDURE GetSize (VAR Symbol : ARRAY OF CHAR; VAR Size : SizeType);
(* determines size of opcode/operand: Byte, Word, Long *)

    VAR
       i : CARDINAL;
       c : CHAR;

    BEGIN
       i := 0;
       REPEAT
          c := Symbol[i];
          INC (i);
       UNTIL (c = 0C) OR (c = '.');

       IF c = 0C THEN
          Size := Word;   (* Default to size Word = 16 bits *)
       ELSE
          c := Symbol[i];   (* Record size extension *)
          Symbol[i - 1] := 0C;   (* Chop size extension off *)
          IF (c = 'B') OR (c = 'S') THEN   (* Byte or Short Branch/Jump *)
             Size := Byte;
          ELSIF c = 'L' THEN
             Size := Long;
          ELSE
             Size := Word;   (* Default size *)
          END;
       END;
    END GetSize;


PROCEDURE GetAbsSize (VAR Symbol : ARRAY OF CHAR; VAR AbsSize : SizeType);
(* determines size of operand: Word or Long *)

    VAR
       i : CARDINAL;
       c : CHAR;
       ParCnt : INTEGER;

    BEGIN
       ParCnt := 0;
       i := 0;
       REPEAT
          c := Symbol[i];
          IF c = '(' THEN
             INC (ParCnt);
          END;
          IF c = ')' THEN
             DEC (ParCnt);
          END;
          INC (i);
       UNTIL (c = 0C) OR ((c = '.') AND (ParCnt = 0));

       IF c = 0C THEN
          AbsSize := Long;
       ELSE
          c := Symbol[i];   (* Record size extension *)
          Symbol[i - 1] := 0C;   (* Chop size extension off *)
          IF (c = 'W') OR (c = 'S') THEN
             AbsSize := Word;
          ELSE
             AbsSize := Long;
          END;
       END;
    END GetAbsSize;



PROCEDURE GetInstModeSize (Mode : OpMode; Size : SizeType;
                           VAR InstSize : CARDINAL) : CARDINAL;
(* Determines the size for the various instruction modes.     *)

    VAR
       n : CARDINAL;

    BEGIN
       CASE Mode OF
          ARDisp,
          ARDisX,
          PCDisp,
          PCDisX,
          AbsW     :  n := 2;
        | AbsL     :  n := 4;
        | MultiM   :  IF Pass2 THEN
                         n := 0;   (* accounted for by code generator *)
                      ELSE
                         n := 2;
                      END;
        | Imm      :  IF Size = Long THEN
                         n := 4;
                      ELSE
                         n := 2;
                      END;
       ELSE
                      n := 0;
       END;

       INC (InstSize, n);
       RETURN (n * 2);
    END GetInstModeSize;


PROCEDURE GetOperand (Oper : OPERAND; VAR Op : OpConfig);
(* Finds mode and value for source or destination operand *)

    VAR
       ch : CHAR;
       C : CARDINAL;   (* holds the ordinal value of a charcter *)
       i, j : CARDINAL;
       Len : CARDINAL;   (* Calculated Length of Oper *)
       TempOp : OPERAND;
       MultFlag : BOOLEAN;

    BEGIN
       Op.Mode := Null;   Op.X := X0;
       Len := Length (Oper);

       IF Len = 0 THEN
          RETURN;
       END;

       GetAbsSize (Oper, AbsSize);

       IF Oper[0] = '#' THEN   (* Immediate *)
          IF Pass2 THEN
             i := 0;
             REPEAT
                INC (i);
                Oper[i - 1] := Oper[i];
             UNTIL Oper[i] = 0C;
             GetValue (Oper, Op.Value);
          END;
          Op.Mode := Imm;
          RETURN;
       END;

       IF Len = 2 THEN   (* possible Addr or Data Register *)
          C := ORD (Oper[1]);
          IF (Oper[0] = 'S') AND (Oper[1] = 'R') THEN
             (* Status Register *)
             Op.Mode := SR;
             RETURN;
          ELSIF (Oper[0] = 'S') AND (Oper[1] = 'P') THEN
             (* Stack Pointer *)
             Op.Mode := ARDir;
             Op.Rn := 7;
             RETURN;
          ELSIF (C >= Zero) AND (C <= Seven) THEN
             (* Looks Like an Addr or Data Reg *)
             IF Oper[0] = 'A' THEN   (* Address Register *)
                Op.Mode := ARDir;
                Op.Rn := C - Zero;
                RETURN;
             ELSIF Oper[0] = 'D' THEN   (* Data Register *)
                Op.Mode := DReg;
                Op.Rn := C - Zero;
                RETURN;
             ELSE
                (* may be a label -- ignore for now *)
             END;
          ELSE
             (* may be a label -- ignore for now *)
          END;
       END;

       IF Len = 3 THEN
          IF (Oper[0] = 'C') AND (Oper[1] = 'C') AND (Oper[2] = 'R') THEN
             (* Condition Code Register *)
             Op.Mode := CCR;
             RETURN;
          ELSIF (Oper[0] = 'U') AND (Oper[1] = 'S') AND (Oper[2] = 'P') THEN
             (* User's Stack Pointer *)
             Op.Mode := USP;
             RETURN;
          ELSE
             (* may be a label -- ignore for now *)
          END;
       END;

       IF (Len = 4) AND (Oper[0] = '(') AND (Oper[3] = ')') THEN
          IF (Oper[1] = 'S') AND (Oper[2] = 'P') THEN
             Op.Mode := ARInd;
             Op.Rn := 7;
             RETURN;
          ELSIF Oper[1] = 'A' THEN
             C := ORD (Oper[2]);
             IF (C >= Zero) AND (C <= Seven) THEN
                Op.Mode := ARInd;
                Op.Rn := C - Zero;
                RETURN;
             ELSE
                Error (Op.Loc, SizeErr);
                RETURN;
             END;
          ELSE
             Error (Op.Loc, AddrErr);
             RETURN;
          END;
       END;

       IF (Len = 5) AND (Oper[0] = '(')
        AND (Oper[3] = ')') AND (Oper[4] = '+') THEN
         (* Address Indirect with Post Inc *)
          IF (Oper[1] = 'S') AND (Oper[2] = 'P') THEN
             (* System Stack Pointer *)
             Op.Mode := ARPost;
             Op.Rn := 7;
             RETURN
          ELSIF Oper[1] = 'A' THEN
             C := ORD (Oper[2]);
             IF (C >= Zero) AND (C <= Seven) THEN
                Op.Mode := ARPost;
                Op.Rn := C - Zero;
                RETURN;
             ELSE
                Error (Op.Loc, SizeErr);
                RETURN;
             END;
```

(continued on page 56)

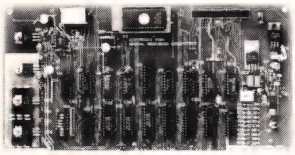
ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

S100 EPROM PROGRAMMER

OUR NEWEST DESIGN, FOR FAST EFFICIENT PROGRAMMING OF THE MOST POPULAR EPROM'S ON YOUR S100 MACHINE. COMES WITH MENU DRIVEN SOFTWARE THAT RUNS UNDER CP/M 2.2 (8 INCH). PC BOARD SET CONSISTS OF (S100) MAIN LOGIC BOARD REMOTE PROGRAMMING CARD AND SIX PERSONALITY MINI BOARDS FOR 2716, 2532, 2732, 2732A, 2764, AND 27128. SOLD AS BARE PC BOARD SET ONLY WITH FULL DOC. SOFTWARE FEATURES "FAST" PROGRAMMING ALGORITHM. FOR Z80 BASED SYSTEMS.



PC BOARD SET, FULL DOCUMENTATION, 8 IN. DISKETTE WITH SOFTWARE.

NEW! $69.95

128K S100 STATIC RAM/EPROM BOARD

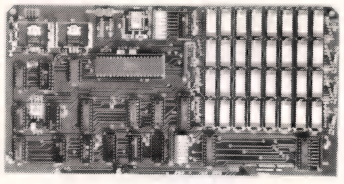
JUST OUT! USES POPULAR 8K X 8 STATIC RAMS (6264) OR 2764 EPROMS. FOR 8 OR 16 BIT DATA TRANSFERS! IEEE 696 STANDARD. LOW POWER. KITS ARE FULLY SOCKETED. FULL DOC AND SCHEMATICS INCLUDED. 24 BIT ADDRESSING.

NEW! $59.95 BARE PC BOARD | $219.00 128K RAM KIT | $139.00 128 EPROM KIT

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



FEATURES:
- 256K on board, using + 5V 64K DRAMS.
- Uses new Intel 8203-1 LSI Memory Controller.
- Requires only 4 Dip Switch Selectable I/O Ports.
- Runs on 8080 or Z80 S100 machines.
- Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- Provisions for Battery back-up.
- Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
- The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $49.95 (8203-1 INTEL $29.95)

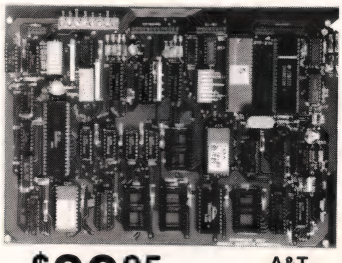
(ADD $50 FOR A&T) #LS-100 $129.00 (FULL 256K KIT)

ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:
- Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- RS232 at 16 BAUD Rates from 75 to 19,200.
- 24 x 80 standard format (60 Hz).
- Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- Composite or Split Video.
- Any polarity of video or sync.
- Inverse Video Capability.
- Small Size: 6.5 x 9 inches.
- Upper & lower case with descenders.
- 7 x 9 Character Matrix.
- Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$89.95 #ZRT-80 (COMPLETE KIT, 2K VIDEO RAM) A&T ADD $50

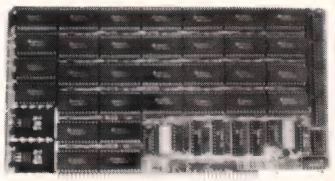
BLANK PCB WITH 2716 CHAR. ROM, 2732 MON. ROM $49.95

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

64K S100 STATIC RAM

$99.00 KIT



LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION $49.95

SUPPORT ICs + CAPS $17.50

FULL SOCKET SET $14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

ASSEMBLED AND TESTED ADD $50

FEATURES: ***PRICE CUT!***
- Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- Fully supports IEEE 696 24 BIT Extended Addressing.
- 64K draws only approximately 500 MA.
- 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- 2716 EPROMs may be installed in any of top 48K.
- Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- BOARD may be partially populated as 56K.

PANASONIC

Green Screen - Video Monitors

25 MHZ. TYPICAL BANDWIDTH!!!

Brand New In The Box! 9-Inch Screen

#K-904B1 (Chassis #Y08A) Open Frame Style

$29.95 EA.

GROUP SPECIAL: **4 for $99.00** WITH DATA & SCHEMATIC

(USA SHIPPING: $3. PER UNIT. CANADA: $7. PER UNIT)

COMPUTER MANUFACTURER'S EXCESS. STILL IN ORIGINAL PANASONIC BOXES. THE CRT TUBE ALONE WOULD COST MORE THAN OUR PRICE FOR THE COMPLETE UNIT. FOR SPLIT VIDEO (TTL INPUTS) OPERATION, NOT COMPOSITE VIDEO. OPERATES FROM 12VDC AT 1 AMP. VERTICAL INPUT IS 49 TO 61 HZ. HORIZONTAL INPUT: 15,750 HZ ± 500 HZ. RESOLUTION IS 800 LINES AT CENTER 650 LINES AT CORNERS.

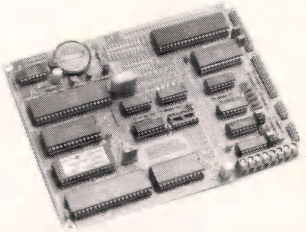
THE NEW 65/9028 VT ANSI VIDEO TERMINAL BOARD!

★ FROM LINGER ENTERPRISES ★

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. Use as a computer console or with a MODEM for hook up to any of the telephone-line computer services.

FEATURES:
- Uses the new SMC 9028 Video Controller Chip coupled with a 6502A CPU.
- RS-232 at 16 Baud Rates from 50 to 19,200
- On board printer port!
- 24 X 80 format (50/60 Hz).
- For 15,750 Hz (Horiz.) monitors.
- 3 Terminal Modes: H-19, ADM3A, and ANSI X 3.64-1979
- Wide and thin-line graphics.
- White characters on black background or reversed.
- Character Attributes: De-Inten, Inverse or Underline.
- Low Power: 5VDC @ .7A, ± 12VDC @ 20MA.
- Mini size: 6.5 X 5 inches.
- Composite or split video.
- 5 X 8 Dot Matrix characters (U/L case).
- Answer back capability.
- Battery backed up status memory.
- For ASCII parallel keyboard.

MICRO SIZE!



$99.95 (Full Kit)

SOURCE DISKETTE: PC/XT FORMAT 5¼ IN. $15

ADD $40 FOR A&T

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

Circle **no. 87** on reader service card.

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

Listing Eighteen (listing continued)

```
      ELSE
         Error (Op.Loc, AddrErr);
         RETURN;
      END;
   END;

   IF (Len = 5) AND (Oper[0] = '-')
    AND (Oper[1] = '(') AND (Oper[4] = ')') THEN
      IF (Oper[2] = 'S') AND (Oper[3] = 'P') THEN
         (* System Stack Pointer *)
         Op.Mode := ARPre;
         Op.Rn := 7;
         RETURN;
      ELSIF Oper[2] = 'A' THEN
         C := ORD (Oper[3]);
         IF (C >= Zero) AND (C <= Seven) THEN
            Op.Mode := ARPre;
            Op.Rn := C - Zero;
            RETURN;
         ELSE
            Error (Op.Loc, SizeErr);
            RETURN;
         END;
      ELSE
         Error (Op.Loc, AddrErr);
         RETURN;
      END;
   END;

   (* Try to split off displacement (if present) *)
   i := 0;
   ch := Oper[i];
   WHILE (ch # '(') AND (ch # 0C) DO    (* move to TempOp *)
      TempOp[i] := ch;
      INC (i);
      ch := Oper[i];
   END;
   TempOp[i] := 0C;    (* Displacement (it it exists) now in TempOp *)

   IF ch = '(' THEN    (* looks like a displacement mode *)
      IF Pass2 THEN
         GetValue (TempOp, Op.Value);    (* Value of Disp. *)
      END;
      j := 0;
      REPEAT    (* put rest of operand (eg. (An,Xi) in TempOp *)
         ch := Oper[i];
         TempOp[j] := ch;
         INC (i);    INC (j);
      UNTIL ch = 0C;
      IF Length (TempOp) > 4 THEN    (* Index may be present *)
         i := 4;    (* Index starts at 4 *)
         j := 0;
         REPEAT                          (* put Xi in Oper *)
            ch := TempOp[i];
            Oper[j] := ch;
            INC (i);    INC (j);
         UNTIL ch = 0C;

         IF Oper[j - 2] = ')' THEN
            Oper[j - 2] := 0C;
         ELSE
            Error (Op.Loc, AddrErr);
            RETURN;
         END;

         GetSize (Oper, Op.Xsize);
         IF Op.Xsize = Byte THEN
            Error (Op.Loc, SizeErr);
            RETURN;
         END;

         C := ORD (Oper[1]);
         IF (Oper[0] = 'S') AND (Oper[1] = 'P') THEN
            (* Stack Pointer *)
            Op.X := Areg;
            Op.Xn := 7;
         ELSIF Oper[0] = 'A' THEN
            IF (C >= Zero) AND (C <= Seven) THEN
               Op.X := Areg;
               Op.Xn := C - Zero;
            ELSE
               Error (Op.Loc, SizeErr);
               RETURN;
            END;
         ELSIF Oper[0] = 'D' THEN
            IF (C >= Zero) AND (C <= Seven) THEN
               Op.X := Dreg;
               Op.Xn := C - Zero;
            ELSE
               Error (Op.Loc, SizeErr);
               RETURN;
            END;
         ELSE
            Error (Op.Loc, AddrErr);
            RETURN;
         END;

         IF (TempOp[1] = 'P') AND (TempOp[2] = 'C') THEN
            Op.Mode :=PCDisX;
            RETURN;
         ELSIF (TempOp[1] = 'S') AND (TempOp[2] = 'P') THEN
            (* Stack Pointer *)
            Op.Rn := 7;
            Op.Mode := ARDisX;
            RETURN;
         ELSIF TempOp[1] = 'A' THEN
            C := ORD (TempOp[2]);
            IF (C >= Zero) AND (C <= Seven) THEN
               Op.Rn := C - Zero;
               Op.Mode := ARDisX;
               RETURN;
            ELSE
               Error (Op.Loc, SizeErr);
               RETURN;
            END;
         ELSE
            Error (Op.Loc, AddrErr);
            RETURN;
         END;
      ELSE    (* No Index *)
         IF (TempOp[1] = 'P') AND (TempOp[2] = 'C') THEN
            Op.Mode := PCDisp;
            RETURN;
         ELSIF (TempOp[1] = 'S') AND (TempOp[2] = 'P') THEN
            (* Stack Pointer *)
            Op.Mode := ARDisp;
            Op.Rn := 7;
            RETURN;
         ELSIF TempOp[1] = 'A' THEN
            C := ORD (TempOp[2]);
            IF (C >= Zero) AND (C <= Seven) THEN
               Op.Rn := C - Zero;
               Op.Mode := ARDisp;
               RETURN;
            ELSE
               Error (Op.Loc, SizeErr);
               RETURN;
            END;
         ELSE
            Error (Op.Loc, AddrErr);
            RETURN;
         END;
      END;
   END;

   (* Check to see if this could be a register list for MOVEM: *)
   i := 0;
   MultFlag := FALSE;
   LOOP
      ch := Oper[i];    INC (i);
      IF ch = 0C THEN
         MultFlag := FALSE;
         EXIT;
      END;
      IF (ch = 'A') OR (ch = 'D') THEN
         ch := Oper[i];    INC (i);    C := ORD (ch);
         IF ch = 0C THEN
            MultFlag := FALSE;
            EXIT;
         END;
         IF (C >= Zero) AND (C <= Seven) THEN
            ch := Oper[i];    INC (i);
            IF ch = 0C THEN
               EXIT
            END;
            IF (ch = '/') OR (ch = '-') THEN
               MultFlag := TRUE;
            END;
         ELSE
            MultFlag := FALSE;
            EXIT;
         END;
      ELSE
         MultFlag := FALSE;
         EXIT;
      END;
   END;
   IF MultFlag THEN
      Op.Mode := MultiM;
      RETURN;
   END;

   (* Must be absolute mode! *)
   IF Pass2 THEN
      GetValue (Oper, Op.Value);
   END;
   IF AbsSize = Word THEN
      Op.Mode := AbsW;
   ELSE
      Op.Mode := AbsL;
   END;
END GetOperand;


PROCEDURE GetMultReg (Oper : OPERAND; PreDec : BOOLEAN;
                      Loc : CARDINAL; VAR MultExt : BITSET);
(* Builds a BITSET marking each register used in a MOVEM instruction *)

   TYPE
      MReg = (D0, D1, D2, D3, D4, D5, D6, D7,
              A0, A1, A2, A3, A4, A5, A6, A7);

   VAR
      i, j : CARDINAL;
      ch : CHAR;
      C : CARDINAL;    (* ORD value of ch *)
      T1, T2 : MReg;    (* Temporary variables for registers *)
      RegStack : ARRAY [0..15] OF MReg;    (* Holds specified registers *)
      SP : CARDINAL;    (* Pointer for Register Stack *)
      RegType : (D, A, Nil);
      Range : BOOLEAN;

   BEGIN
      SP := 0;
      Range := FALSE;
      RegType := Nil;
      i := 0;

      ch := Oper[i];
      WHILE ch # 0C DO
         IF SP > 15 THEN
            Error (Loc, SizeErr);
            RETURN;
         END;

         C := ORD (ch);
         IF ch = 'A' THEN
            IF RegType = Nil THEN
               RegType := A;
            ELSE
               Error (Loc, OperErr);
               RETURN;
            END;
         ELSIF ch = 'D' THEN
            IF RegType = Nil THEN
               RegType := D;
            ELSE
               Error (Loc, OperErr);
               RETURN;
            END;
         ELSIF (C >= Zero) AND (C <= Seven) THEN
            IF RegType # Nil THEN
               T2 := VAL (MReg, (ORD (RegType) * 8) + (C - Zero));
               IF Range THEN
                  Range := FALSE;
                  T1 := RegStack[SP - 1];    (* retreive 1st Reg in range *)
                  FOR j := (ORD (T1) + 1) TO ORD (T2) DO
                     RegStack[SP] := VAL (MReg, j);
                     INC (SP);
                  END;
               ELSE
                  RegStack[SP] := T2;
                  INC (SP);
               END;
            ELSE
               Error (Loc, OperErr);
               RETURN;
```

(continued on page 58)

PROBLEM: There's just no easy way to move from one software program to another.

THE SOFTLOGIC SOLUTION: Software Carousel

Now you can keep up to 10 programs loaded and ready to run.

Hard to believe, but some people are happy with just one kind of PC software. Well, this is not a product for them.

But if you're someone who depends on many packages, all the time—someone who'd use several programs at once if you could, well now you can. With Software Carousel.

With Software Carousel running in RAM, you can load a program and retrieve a file up to 15 times faster. Test conducted on an IBM XT

Why call it "Software Carousel"?

In some ways, Software Carousel works like the slide projector you're used to. You load a handful of pictures, view one at a time, then quickly switch to another. A simple idea, with powerful possibilities for computing.

Here's how it works. When you start Software Carousel, just tell it how much memory you have, load your software and go to work.

Need to crunch numbers? Switch to your spreadsheet. Need your word processor? Don't bother saving your spreadsheet file. Just whip over to your document and do your work. Snap back to your spreadsheet, and it's just like you left it.

With up to ten different programs at your fingertips, you'll have instant access to your database, communications, spelling checker, spreadsheet, word processor, RAM resident utilities, languages, anything you like.

Reach deep into expanded memory.

This could be the best reason ever for owning an expanded memory card, like the Intel Above Board, AST RAMpage, or any card compatible with the L/I/M Extended Memory Standard.

Software Carousel puts programs into this "high-end" memory for temporary storage when they're not in use. And switches them back out when you want them. It's fast, efficient, and easy.

If you want, Software Carousel will even use your hard drive for swapping. Just allocate a portion for storage, and go to work.

Sidekick, Superkey and Ready. All at the same time.

You know what happens if you try loading two or more RAM resident utilities at once. You get crashed keyboards, frozen screens, all kinds of interference between programs fighting for control.

With Software Carousel, you can have as many accessories and utilities on-tap as you want. Just load different ones in different Carousel partitions. Since they can't see each other, they can't fight.

The easy way to maximize PC power.

With all this power, you might think Software Carousel is complicated and difficult to use. Not so. Set it up once, and it will remember forever. Better still, Carousel will look for the programs you use most often, and optimize them for the quickest access.

You can spend a lot more money, and still not get the convenience and productivity increase of Software Carousel.

The way we see it, there are certain things you have the right to expect from your computer. Access to your software is one of them. And at our special introductory price of just $49.95*, Software Carousel is the best way to get it.

But hurry. This price won't last long.

Order today at 800-272-9900 (603-627-9900 in NH) or send the coupon below.

Special combination pricing is available for the purchase of Software Carousel and other SoftLogic products, including DoubleDOS and Disk Optimizer.

Software Carousel $49.95*

YES! Send me_____ copies of Software Carousel at the Special Introductory Price of just $49.95.

Name_____

Company_____

Address_____

City_____ State/Zip_____

Check Enclosed ☐ VISA ☐ MC ☐ AMEX ☐

Card # Exp. Date

Signature_____

SoftLogic Solutions, Inc.
530 Chestnut Street
Manchester, NH 03101
800-272-9900
(603-627-9900 in NH)

SOFTLOGIC SOLUTIONS

Call today: 800-272-9900

*plus $5.00 shipping and handling.

Circle **no. 284** on reader service card.

Listing Eighteen (listing continued)

```
            END;
         ELSIF ch = '-' THEN
            IF (Range = FALSE) AND (RegType # Nil) AND (i > 0) THEN
               RegType := Nil;
               Range := TRUE;
            ELSE
               Error (Loc, OperErr);
               RETURN;
            END;
         ELSIF ch = '/' THEN
            IF (Range = FALSE) AND (RegType # Nil) AND (i > 0) THEN
               RegType := Nil;
            ELSE
               Error (Loc, OperErr);
               RETURN;
            END;
         ELSE
            Error (Loc, OperErr);
            RETURN;
         END;

         INC (i);
         ch := Oper[i];
      END;

      MultExt := {};
      FOR j := 0 TO SP - 1 DO
         C := ORD (RegStack[j]);
         IF PreDec THEN
            C := 15 - C;
         END;
         INCL (MultExt, C);
      END;
   END GetMultReg;

END SyntaxAnalyzer.
```

End Listing Eighteen

Listing Nineteen

```
IMPLEMENTATION MODULE Listing;
(* Creates a program listing, including Addresses, Code & Source. *)

   FROM Files IMPORT
      FILE, Write;

   FROM LongNumbers IMPORT
      LONG, LongPut;

   FROM Parser IMPORT
      TOKEN, Line;

   FROM SymbolTable IMPORT
      ListSymTab;

   FROM Conversions IMPORT
      CardToStr;

   IMPORT ASCII;


   CONST
      LnMAX = 55;


   VAR
      LnCnt : CARDINAL;    (* counts number of lines per page *)
      PgCnt : CARDINAL;    (* count of page numbers *)


   PROCEDURE WriteStrF (f : FILE; Str : ARRAY OF CHAR);
   (* Writes a string to the file *)

      VAR
         i : CARDINAL;

      BEGIN
         i := 0;
         WHILE Str[i] # 0C DO
            Write (f, Str[i]);
            INC (i);
         END;
      END WriteStrF;


   PROCEDURE CheckPage (f : FILE);
   (* Checks if end of page reached yet -- if so, advances to next page. *)

      VAR
         i : CARDINAL;
         PgCntStr : ARRAY [0..6] OF CHAR;

      BEGIN
         INC (LnCnt);
         IF LnCnt >= LnMAX THEN
            LnCnt := 1;
            INC (PgCnt);
            Write (f, ASCII.ff);    (* Form Feed for new page *)
            IF CardToStr (PgCnt, PgCntStr) THEN    (* Print New Page Number *)
               FOR i := 1 TO 60 DO
                  Write (f, ' ');
               END;

               WriteStrF (f, "Page ");
               WriteStrF (f, PgCntStr);
            END;
            FOR i := 1 TO 3 DO
               Write (f, ASCII.cr);
               Write (f, ASCII.lf);
            END;
         END;
      END CheckPage;


   PROCEDURE StartListing (f : FILE);
   (* Sign on messages for listing file -- initialize *)
      BEGIN
         Write (f, ASCII.ff);    (* Start on a clean page *)

         WriteStrF (f, "                    68000 Cross Assembler");
         Write (f, ASCII.cr);
         Write (f, ASCII.lf);

         WriteStrF (f, "          Copyright (c) 1985 by Brian R. Anderson");
         Write (f, ASCII.cr);
         Write (f, ASCII.lf);

         Write (f, ASCII.cr);
         Write (f, ASCII.lf);

         LnCnt := 1;
         PgCnt := 1;
      END StartListing;


   PROCEDURE WriteListLine (f : FILE;
                            AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
                               nA,     nO,     nS,     nD    : CARDINAL);
   (* Writes one line to the Listing file, including Object Code *)

      CONST
         ObjMAX = 30;

      VAR
         i : CARDINAL;

      BEGIN
         IF nA = 0 THEN    (* nA is always either 0 or 6.  Address field = 8 *)
            FOR i := 1 TO 8 DO
               Write (f, ' ');
            END;
         ELSE
            LongPut (f, AddrCnt, 6);
            Write (f, ' ');
            Write (f, ' ');
         END;
         LongPut (f, ObjOp, nO);
         LongPut (f, ObjSrc, nS);
         LongPut (f, ObjDest, nD);
         i := 8 + nO + nS + nD;
         WHILE i < ObjMAX DO
            Write (f, ' ');
            INC (i);
         END;



         WriteStrF (f, Line);
         Write (f, ASCII.cr);
         Write (f, ASCII.lf);

         CheckPage (f);

      END WriteListLine;


   PROCEDURE WriteSymTab (f : FILE; NumSym : CARDINAL);
   (* Lists symbol table in alphabetical order *)

      VAR
         Label : TOKEN;
         Value : LONG;
         i : CARDINAL;

      BEGIN
         LnCnt := 1;
         INC (PgCnt);

         WriteStrF (f, "              * * *  Symbolic Reference Table  * * *");
         FOR i := 1 TO 3 DO
            Write (f, ASCII.cr);
            Write (f, ASCII.lf);
         END;

         FOR i := 1 TO NumSym DO
            ListSymTab (i, Label, Value);
            WriteStrF (f, Label);
            WriteStrF (f, "  :  ");
            LongPut (f, Value, 8);
            Write (f, ASCII.cr);
            Write (f, ASCII.lf);
            CheckPage (f);
         END;

         Write (f, ASCII.ff);
      END WriteSymTab;
```

End Listing Nineteen

Listing Twenty

```
IMPLEMENTATION MODULE Srecord;
(* Creates Motorola S-records of program:          *)
(*    S0 = header record,                          *)
(*    S2 = code/data records (24 bit address),     *)
(*    S8 = termination record (24 bit address).    *)

   FROM Files IMPORT
      FILE, Write;

   FROM Strings IMPORT
      Length;
```

(continued on page 60)

PROBLEM: Handling your need for more megabytes, without spending megabucks on a new drive.

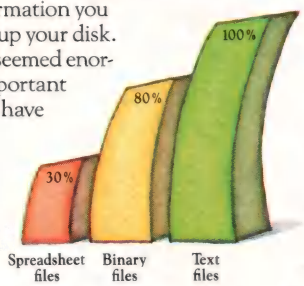
THE SOFTLOGIC SOLUTION: Cubit™

Now get up to twice the capacity from all your storage media.

You know what happens. The more you use your computer, the more information you create. And the faster you fill up your disk.

The 10MB drive that once seemed enormous is now jammed with important files. That 20MB that should have lasted years is crowded in a matter of months.

Of course you could keep buying bigger hard drives. Or you could get Cubit and get the maximum storage space from the drives you already have.



With Cubit, you'll get as much as 100% compression on data files, effectively doubling the storage capacity of all your magnetic media.

What is Cubit?

In brief, Cubit is an advanced software tool that automatically reduces the number of bytes required to store a file, then converts the file back to its original size when retrieved. Some programmers call this effect "data compression," others, "disk expansion." Either way, the result is the same.

Here's how it works. When Cubit compresses a file, it first compares each word to its massive English word dictionary. Words that match are reduced to a predetermined code of just one, two or three bytes each. It then saves the abbreviated version to disk. Decompression works just the opposite.

To accommodate other words and symbols, Cubit uses two more compression techniques. One assigns new, shorter codes to unusual words. Another compresses according to the frequency of character strings in non-text data. So no matter what kind of files you create, Cubit ensures maximum space savings.

Best of all, you'll be using the same fast, reliable data compression techniques used on mainframe computers for decades.

How much disk space will you save?

Because the vast majority of data created on PC's is standard ASCII text—letters, numbers and other English language symbols—we've optimized Cubit for word processing and database files. With these, you'll get a minimum of 50% expansion on up to a full 100% or more.

At the same time, you can expect a significant 30% to 50% improvement with other kinds of data. Including spreadsheet files, program code, graph and image files, even binary data.

And Cubit works just as well with floppies and tape cassettes as it does with hard disk drives.

Run Cubit where you want, when you want.

Maybe you'll want to use Cubit for all your files, or maybe just some. So Cubit lets you specify exactly which files to work on and which ones to leave alone.

In RAM resident mode, Cubit works quickly and invisibly, compressing and decompressing right from within any program you run. Or use Cubit's powerful file management mode. It supports wild-card and global file names, and addresses sub-directories up to thirty levels deep.

Save time and money, as well as disk space.

A compressed file is a smaller file. So with Cubit, back-ups take less time, as well as less space. And communicating compressed files means significant savings on phone line charges.

Any way you look at it, Cubit will pay for itself in no time. And that's especially true now.

Special limited time offer.

Buy Cubit now and you'll save even more. Because for a limited time, you can buy Cubit at the special introductory price of just $49.95.* But hurry. This special price won't last long.

Ask for Cubit at your computer dealer. Or order directly from SoftLogic Solutions by calling 800-272-9900 (603-627-9900 in NH), or mail in the coupon below.

Special pricing is available when you buy Cubit along with other SoftLogic products including DoubleDOS, Software Carousel and Disk Optimizer. Ask for details.

Cubit™ $49.95*

YES! Please send me____copies of Cubit at this special introductory price.

Name_____

Company_____

Address_____

City_____ State/Zip_____

Check Enclosed ☐ VISA ☐ MC ☐ AMEX ☐

Card # Exp. Date

Signature_____

SoftLogic Solutions, Inc.
530 Chestnut Street
Manchester, NH 03101
800-272-9900
(603-627-9900 in NH)

SOFTLOGIC SOLUTIONS

Call today: 800-272-9900

*plus $5.00 shipping and handling.

Circle **no. 293** on reader service card.

Listing Twenty (listing continued)

```
FROM LongNumbers IMPORT
   LONG, LongAdd, LongSub, LongInc, LongDec, LongClear,
   LongCompare, CardToLong, LongPut;

IMPORT ASCII;


CONST
   CountMAX = 16;
   SrecMAX = CountMAX * 2;
   XrecMAX = SrecMAX;


VAR
   StartAddr : LONG;    (* address that record starts on *)
   TempAddr : LONG;     (* running address of where we are now *)
   CheckSum : LONG;
   Count : CARDINAL;    (* count of HEX-pairs in S-record *)
   Sdata : ARRAY [1..SrecMAX] OF INTEGER;    (* S-record data, HEX digits *)
   Sindex : CARDINAL;    (* index for Sdata array *)
   Xdata : ARRAY [1..XrecMAX] OF INTEGER;    (* Overflow for Sdata *)
   Xindex : CARDINAL;    (* index for Xdata array *)
   Boundary : BOOLEAN;    (* marks Address MOD 16 boundary of S-record *)
   LZero : LONG;    (* used as a constant = 0 *)


PROCEDURE Complement;    (* CheckSum *)
   BEGIN
      LongSub (LZero, CheckSum, CheckSum);    (* 2's Complement *)
      LongDec (CheckSum, 1);    (* Make it 1's Complement *)
   END Complement;


PROCEDURE AppendSdata (Data : LONG; n : CARDINAL) : BOOLEAN;
(* Transfers data to Sdata, and updates Count & CheckSum. *)
(*    If no room: Data goes to Xdata & FALSE returned.    *)

   VAR
      T : LONG;    (* temporary -- used only as a 2 digit HEX number *)

   BEGIN
      T := LZero;

      WHILE (n # 0) AND (Count # CountMAX) AND (NOT Boundary) DO
         Sdata[Sindex] := Data[n];
         Sdata[Sindex - 1] := Data[n - 1];

         T[2] := Data[n];    T[1] := Data[n - 1];
         LongAdd (T, CheckSum, CheckSum);

         DEC (n, 2);
         DEC (Sindex, 2);
         INC (Count);

         LongInc (TempAddr, 1);
         IF TempAddr[1] = 0 THEN    (* i.e., TempAddr MOD 16 = 0 *)
            Boundary := TRUE;
         END;
      END;

      IF (Count = CountMAX) OR (Boundary) THEN
         WHILE n > 0 DO    (* Add Data to Xdata (in reverse) *)
            INC (Xindex);
            Xdata[Xindex] := Data[n];
            DEC (n);
         END;

         RETURN FALSE;    (* Sdata is full *)
      ELSE
         RETURN TRUE;
      END;
   END AppendSdata;


PROCEDURE DumpSdata (f : FILE);
(* Writes an S2 record to the file *)

   VAR
      T : LONG;    (* temporary -- used to output Count & CheckSum *)
      i, j : CARDINAL;

   BEGIN
      IF Count = 0 THEN
         RETURN;    (* nothing to dump *)
      END;

      Write (f, 'S');
      Write (f, '2');

      CardToLong (Count + 4, T);    (* extra for Address & Checksum *)
      LongPut (f, T, 2);
      LongAdd (T, CheckSum, CheckSum);    (* Add Count to CheckSum *)

      LongPut (f, StartAddr, 6);
      (* Add Address to CheckSum *)
      T := LZero;
      T[1] := StartAddr[1];    T[2] := StartAddr[2];
      LongAdd (T, CheckSum, CheckSum);
      T[1] := StartAddr[3];    T[2] := StartAddr[4];
      LongAdd (T, CheckSum, CheckSum);
      T[1] := StartAddr[5];    T[2] := StartAddr[6];
      LongAdd (T, CheckSum, CheckSum);

      IF Count < CountMAX THEN    (* adjust short record -- shuffle down *)
         j := 1;
         FOR i := Sindex + 1 TO SrecMAX DO
            Sdata[j] := Sdata[i];
            INC (j);
         END;
      END;
      LongPut (f, Sdata, Count * 2);    (* S-record Code/Data *)

      Complement; (* CheckSum *)
      LongPut (f, CheckSum, 2);



      Write (f, ASCII.cr);
      Write (f, ASCII.lf);

      LongInc (StartAddr, Count);
      Sindex := SrecMAX;
      Count := 0;
      Boundary := FALSE;
      CheckSum := LZero;
   END DumpSdata;


PROCEDURE GetXdata;
(* Transfer Xdata into new Sdata line -- N.B.: Xdata stored in reverse *)

   VAR
      i : CARDINAL;
      T : LONG;

   BEGIN
      i := 1;
      T := LZero;

      (* No need for either of the tests (CountMAX or Boundary)    *)
      (* used in AppendSdata.  GetXdata is only ever called        *)
      (* after DumpSdata and is therefore only putting (up to 20)  *)
      (* HEX digits in an empty buffer (which could hold 32).      *)
      WHILE i < Xindex DO
         Sdata[Sindex] := Xdata[i];
         Sdata[Sindex - 1] := Xdata[i + 1];
         T[2] := Sdata[Sindex];    T[1] := Sdata[Sindex - 1];
         LongAdd (T, CheckSum, CheckSum);
         INC (i, 2);
         DEC (Sindex, 2);
         INC (Count);
         LongInc (TempAddr, 1);
      END;

      Xindex := 0;
   END GetXdata;


PROCEDURE StartSrec (f : FILE; SourceFN : ARRAY OF CHAR);
(* Writes S0 record (HEADER) and initializes *)

   VAR
      T : LONG;    (* temporary *)
      i : CARDINAL;

   BEGIN
      Write (f, 'S');
      Write (f, '0');

      CheckSum := LZero;
      Count := Length (SourceFN) + 3;    (* extra for Address & Checksum *)
      CardToLong (Count, T);
      LongPut (f, T, 2);
      LongAdd (T, CheckSum, CheckSum);

      LongPut (f, LZero, 4);    (* Address is 4 digit, all zero, for S0 *)

      i := 0;
      WHILE SourceFN[i] # 0C DO
         CardToLong (ORD (SourceFN[i]), T);
         LongAdd (T, CheckSum, CheckSum);
         LongPut (f, T, 2);
         INC (i);
      END;

      Complement;    (* CheckSum *)
      LongPut (f, CheckSum, 2);

      Write (f, ASCII.cr);
      Write (f, ASCII.lf);

      Sindex := SrecMAX;
      Xindex := 0;
      Count := 0;
      Boundary := FALSE;
      CheckSum := LZero;
      StartAddr := LZero;
      TempAddr := LZero;
   END StartSrec;


PROCEDURE WriteSrecLine (f : FILE;
                         AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
                            nA,     nO,     nS,      nD    : CARDINAL);
(* Collects Object Code -- Writes an S2 record to file if line is full *)

   VAR
      dummy : BOOLEAN;

   BEGIN
      IF nA = 0 THEN
         RETURN;    (* Nothing to add to S-record *)
      END;

      IF Xindex # 0 THEN
         GetXdata;    (* transfers Xdata into Sdata *)
      END;

      IF LongCompare (AddrCnt, TempAddr) # 0 THEN
         DumpSdata (f);
      END;

      IF Count = 0 THEN
         StartAddr := AddrCnt;
         TempAddr := AddrCnt;
      END;

      dummy := AppendSdata (ObjOp, nO);
      dummy := AppendSdata (ObjSrc, nS);
      IF NOT AppendSdata (ObjDest, nD) THEN
         DumpSdata (f);
      END;
   END WriteSrecLine;


PROCEDURE EndSrec (f : FILE);
(* Finishes off any left-over (Partial) S2 line, *)
(* and then writes S8 record (TRAILER)            *)
   BEGIN
      IF Xindex # 0 THEN
         GetXdata;



         END;
         DumpSdata (f);

         Write (f, 'S');    (* Fixed format for S8 record *)
         Write (f, '8');
         Write (f, '0');
         Write (f, '4');
         Write (f, '0');
         Write (f, '0');
         Write (f, '0');
```

(continued on page 62)

PROBLEM: The more experience your hard disk has, the harder it has to work.

THE SOFTLOGIC SOLUTION: Disk Optimizer™

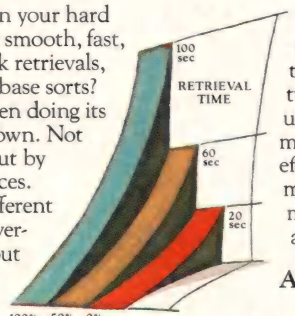
Your hard disk will run faster when it's not chasing around after files.

Remember the old days when your hard drive was new? Remember that smooth, fast, slick performance? Those quick retrievals, rapid saves, lightning-like database sorts?

Well ever since, DOS has been doing its best to slow your hard drive down. Not by slowing down the motor, but by breaking your files up into pieces. Storing different chunks in different places. Data files, programs, overlays and batches that started out in one seamless piece are now scattered all over.

Loading is slower. Sorting is slower. Retrieving, backing-up. Everything takes longer because your disk has to work harder.

Problem is, it's something that happens so gradually you may not notice the difference. At least, not until you see the dramatic improvement after using Disk Optimizer.



The more fragmented your files get, the longer they take to retrieve.

File fragmentation—It's a problem you can see.

Watch your hard drive the next time it reads or writes a file. Each "blip" of the LED means the drive-head is moving to another place on the disk—either to pick up or lay down another chunk of data.

And the truth is, head movement takes time. Far more time than actual reading and writing. What's worse, all this head movement causes extra wear and tear that can shorten the life of your drive.

Disk Optimizer—Tunes up your disk by cleaning up your files.

Disk Optimizer works by finding all the scattered pieces of your files and putting them back together where they belong. Next time your drive reads it, there's just one place to look.

And the results are often dramatic. Reading and writing times may be cut by as much as two thirds. Database sorts that used to take hundreds of head moves now proceed quickly and efficiently. And since head movement is now at an absolute minimum, your disk drive will lead a longer, more productive life.

Analyze, scrutinize, optimize.

Before you optimize, you'll probably want to analyze. So Disk Optimizer shows you, in percentages, how much fragmentation has taken place—on the entire disk, in individual directories, or for groups of files you specify using global or wildcard names.

Plus, there's built-in data security that lets you assign passwords to as many files or file groups as you want.

And the File Peeker gives you an inside look at the structure of files. It's a great way for non-programmers to learn more about computers, and a powerful tool for professionals who want to analyze the contents of their disks.

Get your hard drive back in shape—at a special low price.

When you think about it, it's simple. The longer you own your hard drive, the more you come to depend on it. But the longer you wait to get Disk Optimizer, the less performance you'll have.

And the less chance you'll have to buy Disk Optimizer at the special introductory price of just $49.95*.

That's a small price to pay to get back the speed you depend on. But it's a price that won't last long.

Ask for Disk Optimizer at your computer dealer.

Or order today by calling SoftLogic Solutions at 800-272-9900 (603-627-9900 in NH), or send the coupon below.

Special combination pricing is available when you buy Disk Optimizer along with other SoftLogic products like DoubleDOS, Software Carousel and Cubit.

Ask for details.

Disk Optimizer $49.95*

YES! Please send me_____copies of Disk Optimizer at this special introductory price.

Name_____

Company_____

Address_____

City_____ State/Zip_____

Check Enclosed ☐ VISA ☐ MC ☐ AMEX ☐

Card #_____ Exp. Date_____

Signature_____

SoftLogic Solutions, Inc.
530 Chestnut Street
Manchester, NH 03101
800-272-9900
(603-627-9900 in NH)

SOFTLOGIC SOLUTIONS

Call today: 800-272-9900

*plus $5.00 shipping and handling.

Circle **no. 163** on reader service card.

Listing Twenty (listing continued)

```
         Write (f, '0');
         Write (f, '0');
         Write (f, '0');
         Write (f, 'F');
         Write (f, 'C');
         Write (f, ASCII.cr);
         Write (f, ASCII.lf);
         Write (f, ASCII.cr);
         Write (f, ASCII.lf);
      END EndSrec;

BEGIN   (* Initialization *)
   LongClear (LZero);
END Srecord.
```

End Listing Twenty

Listing Twenty-One

```
IMPLEMENTATION MODULE ErrorX68;
(* Displays error messages for X68000 cross assembler *)

   FROM Terminal IMPORT
      WriteString, WriteLn;

   IMPORT Terminal;   (* for Read/Write *)

   FROM Files IMPORT
      FILE;

   IMPORT Files;   (* for Write *)

   FROM Strings IMPORT
      Length;

   FROM Conversions IMPORT
      CardToStr;

   IMPORT ASCII;

   FROM Parser IMPORT
      Line, LineCount;


(*---
   TYPE
      ErrorType = (Dummy, TooLong, NoCode, SymDup, Undef, SymFull, Phase,
                   ModeErr, OperErr, BraErr, AddrErr, SizeErr, EndErr);

   VAR
      ErrorCount : CARDINAL;
                                                                    ---*)

   VAR
      FirstTime : BOOLEAN;


   PROCEDURE FileWriteString (f : FILE; VAR Str : ARRAY OF CHAR);

      VAR
         i : CARDINAL;

      BEGIN
         i := 0;
         WHILE Str[i] # 0C DO
            Files.Write (f, Str[i]);
            INC (i);
         END;
      END FileWriteString;


   PROCEDURE Error (Pos : CARDINAL; ErrorNbr : ErrorType);
   (* Displays Error #ErrorNbr, then waits for any key to continue *)

      VAR
         i : CARDINAL;
         c : CHAR;
         CntStr : ARRAY [0..6] OF CHAR;
         dummy : BOOLEAN;

      BEGIN
         WriteLn;
         dummy := CardToStr (LineCount, CntStr);
         WriteString (CntStr);
         WriteString ("    ");
         WriteString (Line);   WriteLn;

         (* Make up for LineCnt so ^ in right spot *)
         FOR i := 1 TO Length (CntStr) DO
            Terminal.Write (' ');
         END;
         WriteString ("    ");

         IF Pos > 0 THEN
            FOR i := 1 TO Pos DO
               Terminal.Write (' ');
            END;
            Terminal.Write ('^');   WriteLn;
         END;

         CASE ErrorNbr OF
            TooLong : WriteString ("Identifier too long -- Truncated!");
         |  NoCode  : WriteString ("No such op-code.");
         |  SymDup  : WriteString ("Duplicate Symbol.");
         |  Undef   : WriteString ("Undefined Symbol.");
         |  SymFull : WriteString ("Symbol Table Full -- Maximum = 500!");
                      WriteLn;
                      WriteString ("Program Terminated.");   WriteLn;
                      HALT;
         |  Phase   : WriteString ("Pass 1/Pass 2 Address Count Mis-Match.");
         |  ModeErr : WriteString ("This addressing mode not allowed here.");
         |  OperErr : WriteString ("Error in operand format.");
         |  BraErr  : WriteString ("Error in relative branch.");
         |  AddrErr : WriteString ("Address mode error.");
         |  SizeErr : WriteString ("Operand size error.");
         |  EndErr  : WriteString ("Missing END Pseudo-Op.");
         ELSE
            WriteString ("Unknown Error.");
         END;
         WriteLn;

         IF FirstTime THEN
            WriteString ("Hit any key to continue.... ");
            Terminal.Read (c);
            WriteLn;
            FirstTime := FALSE;
         ELSE
            Terminal.Read (c);
         END;

         INC (ErrorCount);
         IF ErrorCount > 500 THEN
            WriteString ("Too many errors!");   WriteLn;
            WriteString ("Program Terminated.");   WriteLn;
            HALT;
         END;
      END Error;


PROCEDURE WriteErrorCount (f : FILE);
(* Error count output to Console & Listing file *)

   VAR
      CntStr : ARRAY [0..6] OF CHAR;
      Msg0 : ARRAY [0..25] OF CHAR;

         Msg1 : ARRAY [0..10] OF CHAR;
         Msg2 : ARRAY [0..20] OF CHAR;
         dummy : BOOLEAN;

      BEGIN
         Msg0 := "--->   END OF ASSEMBLY";
         Msg1 := "--->    ";
         Msg2 := " ASSEMBLY ERROR(S).";
         dummy := CardToStr (ErrorCount, CntStr);

         (* Messages to console *)
         WriteLn;
         WriteLn;
         WriteString (Msg0);   WriteLn;
         WriteString (Msg1);
         WriteString (CntStr);
         WriteString (Msg2);
         WriteLn;

         (* Messages to listing file *)
         Files.Write (f, ASCII.cr);
         Files.Write (f, ASCII.lf);
         Files.Write (f, ASCII.cr);
         Files.Write (f, ASCII.lf);

         FileWriteString (f, Msg0);
         Files.Write (f, ASCII.cr);
         Files.Write (f, ASCII.lf);

         FileWriteString (f, Msg1);
         FileWriteString (f, CntStr);
         FileWriteString (f, Msg2);
         Files.Write (f, ASCII.cr);
         Files.Write (f, ASCII.lf);

         Files.Write (f, ASCII.ff);   (* feed up next page *)
      END WriteErrorCount;

BEGIN   (* MODULE Initialization *)
   FirstTime := TRUE;
   ErrorCount := 0;
END ErrorX68.
```

End Listings

M A K E • T H E • C O N N E C T I O N

Lattice

Lattice is a famous name in the world of high quality programmer's tools. As an authorized distributor of Lattice products, we at Programmer's Connection are pleased to be your source for their complete line of IBM Personal Computer software. All of these products come with full manufacturer's support directly from Lattice, Incorporated.

We specialize in development tools for IBM Personal Computers and take pride in providing our customers with the finest in selection, availability, price and service.

		LIST	OURS
Lattice C Compiler		500	299
	with Library Source Code	900	549

Popular, industry standard C compiler that features fast compilation, efficient code generation, support for 80186/286 instructions and inline support of 8087/287 instructions. The latest version now supports void, enum, unsigned as a modifier and function prototype checking. The library contains more than 325 functions compatible with UNIX, XENIX and the proposed ANSI standard, plus extensive support for MS-DOS versions 2.+ and 3.+. Other useful Lattice C features include support for nested comments, extended symbol length and multiple memory models. It comes with an object module disassembler, a function extract utility, a full set of libraries for each supported memory model, sample programs and extensive documentation. Requires 128K memory.

The C-Food Smorgasbord	Binary	150	99
	with Source Code	300	195

General C function library featuring BCD (binary coded decimal) arithmetic, level 0 I/O, BIOS interface, terminal independence, directory, clock, string and other miscellaneous functions. No royalties. For use with Lattice C.

C-SPRITE		175	139

Program debugger with source level support for Lattice C that includes help screens, macros, command files, conditional commands, debugging through a COM port and support for Plink86 overlays. The source mode supports all debugging functions including disassemble, single-step and breakpoints. The data types of symbols may be completely specified so that variables can be properly displayed. There is also complete assembly language support providing direct access to machine addresses and instructions. Requires 256K memory. Specify C compiler: Lattice or Microsoft.

Curses Screen Manager	Binary	125	99
	with Source Code	250	199

Library of C screen interface functions compatible with curses packages on UNIX systems. You can keep and update any number of full or partial virtual screen images in memory and display them as needed. Functions are provided to write text to virtual screens, move the cursors, scroll the screens, overlay screens, outline, insert, delete, clear and highlight. No royalties. For use with Lattice C.

dBC II or dBC III	Binary	250	199
	with Source Code	500	395

Complete C library of ISAM file management functions for creating and manipulating dBase compatible files. You can easily add, update, delete, retrieve and organize records and indexes in dBase format. Up to eight data and eight index files may be opened and processed simultaneously. Specify dBC II for dBase II type files or dBC III for dBase III type files. No royalties. Requires 128K memory. Specify C compiler: Lattice, Microsoft, Computer Innovations or DeSmet.

LMK Make Utility		195	149

Programming utility to rebuild programs after changes have been made to source files. First, you create a text file consisting of macro definitions, dependency descriptions and executable commands. Then, whenever you make changes to your program, LMK determines which source files need to be recompiled and automatically creates the new program. Requires 128K memory and may be used with any compiler or assembler.

LSE Screen Editor		125	99

Multi-window programmer's editor with block moves, pattern searching and "cut and paste." You can remap any of LSE's 48 keystrokes to suit your own preferences and define your own keyboard macros and default file extensions. The menus, prompts and help messages used in the system can all be customized. Special features include a Lattice C error tracking mode and three assembly language input modes. Requires 128K memory.

RPG II Compiler		750	595

RPG II compiler for MS-DOS that is compatible with IBM System III, System/34 and System/36 RPG II compilers. Special PC extensions include support for standard MS-DOS files, keyboard, function keys and string handling. ISAM files are compatible with dBC III and dBase III files. Requires 192K memory.

SecretDisk		60	49

File security utility for providing complete security for sensitive information on a floppy or hard disk system. You can use either the international Data Encryption Standard (DES) or Lattice's own Fast encryption algorithm for higher speed operation. It's loaded as a DOS device driver and creates new logical DOS drives where all files are fully encrypted. A password is entered when the system is booted and protection can be switched on and off with a single password controlled command line. Without the password, there is no known way to access the encrypted files! Multiple protected areas may be created using different passwords and data backup may be made in either encrypted or unencrypted mode. It does not interfere with normal access to the computer system or to files that are not encrypted.

SideTalk		120	95

Pop-up telecommunications package that can be accessed from inside any application with a single keystroke. It incorporates the SideTalk Communications Language (SCL) consisting of BASIC-like commands that allow you to create your own communications processing system. It provides for multitasking (background) operation, file transfer capabilities, text transfer from background to foreground and DOS commands available in background. Requires less than 64K available memory.

Text Management Utilities		120	95

Includes four text management utilities found under UNIX. The first utility is grep (global regular expression search and print). You provide it with a pattern to find and it displays each line containing that pattern with its line number in that file. In addition, these functions are provided as Lattice C object libraries. The second utility is DIFF, a differential file comparator. It compares two files and determines how they differ from one another. The third utility is ED, a line editor and the fourth utility is WC, a simple word count facility for counting the number of characters, words and lines in a file. Requires 128K memory.

TopView Toolbasket	Binary	250	199
	with Source Code	500	395

Library of C functions for simplifying programming in IBM's TopView environment. It gives you easy access to TopView's window, cursor, pointer facilities, cut-and-paste services and printer control services. It deals with TopView objects through a central dispatching function that can be tailored to your application. Includes excellent error checking and debugging support. Requires 256K memory (512K recommended). For use with Lattice C.

CALL TOLL FREE

800·336·1166
U.S. OHIO 216-877-3781

800·225·1166
CANADA

programmer's connection

Lattice is a trademark of Lattice, Incorporated.

Turn the page for a wide selection of programmer's development tools exclusively for IBM Personal Computers and compatibles.

Circle **no. 129** on reader service card.

PROGRAMMER DEVELOPMENT TOOLS FOR THE IBM-PC/XT/AT and compatibles.

NO *Shipping Charge**
Handling Charge
Insurance Charge
Credit Card Charge
C.O.D. Charge
Purchase Order Charge
Hidden Charges
Risk Guarantee

***When shipping via standard United Parcel Service.**

apl language

		LIST	OURS
APL*PLUS/PC System by STSC		595	**449**
APL*PLUS/PC Tools Vol 1 by STSC		295	**239**
APL*PLUS/PC Tools Vol 2 by STSC		150	**129**
APL*PLUS/UNX System by STSC	*For AT Xenix*	995	**795**
Btrieve *ISAM File Mgr* by SoftCraft	*New version*	250	**195**
Financial/Statistical Library by STSC		275	**219**
FRESCO *Business Graphics System* by Mr. APL		300	**269**
Pocket APL by STSC		95	**79**
STATGRAPHICS by STSC		695	**539**

artificial intelligence

ExpertEASE by Human Edge	*New*	695	**589**
ExpertEDGE by Human Edge	*New*	795	**659**
Experteach by Intelliware	*Complete System*	475	**389**
EXSYS *Expert System Development Software*		395	**339**
GCLISP *Golden Common LISP* by Gold Hill	*All models*	CALL	**CALL**
Insight I by Level Five Research	*AI Primer*	95	**75**
Insight II by Level Five Research		485	**389**
LISP by Microsoft	*New Common Lisp*	250	**189**
Methods *Smalltalk-based Prototyping* by Digitalk		250	**209**
MicroProlog by Programming Logic Associates	*New*	250	**219**
with APES		425	**369**
Professional MicroProlog by Programming Logic	*New*	395	**349**
with APES		650	**569**
Prolog-86 from Solution Systems		125	**CALL**
Prolog-86 Plus from Solution Systems	*New*	250	**CALL**
QNIAL by NIAL Systems	*New*	375	**359**
Small-X by Kaplan		125	**99**
TransLISP from Solution Systems	*New*	75	**CALL**
Turbo Prolog Compiler by Borland International	*New*	100	**CALL**

assemblers and debuggers

8088 Assembler *w/Z-80 Translator* by 2500 AD		100	**89**
Advanced Trace-86 by Morgan	*with ASM Interpreter*	175	**139**
Codesmith-86 *Debugger* by Visual Age		145	**109**
Cross Assemblers from 2500AD	*Over 25 varieties*	CALL	**CALL**
Microsoft Macro Assembler with utilities		150	**99**
Periscope I *Debugger* by Data Base Decisions		295	**249**
Periscope II by Data Base Decisions		145	**115**
The PROFILER by DWB Associates	*Includes source*	125	**95**
Turbo EDITASM *Fast Assembler* by Speedware		99	**84**
Visible Computer: 8088 by Software Masters		80	**65**

basic language

BetterBASIC by Summit Software	*Now BASICA Compatible*	200	**165**
8087 Math Support		99	**85**
Btrieve Interface	*Used with Btrieve*	99	**85**
C Interface	*New*	CALL	**CALL**
Run-time Module		250	**225**
Microsoft QuickBASIC Compiler	*BASICA Compatible*	99	**79**
Professional BASIC by Morgan		99	**79**
8087 Math Support		50	**47**
True Basic from Addison-Wesley		150	**105**
Run-time Module		500	**435**

blaise products

Asynch Manager *for C or Pascal*		175	**139**
C Tools		125	**105**
C Tools 2		100	**84**
Combination package	*Special Price*	175	**149**
Exec *Program Chainer*		95	**79**
Pascal Tools		125	**105**
Pascal Tools 2		100	**84**
Turbo ASYNCH *for Turbo Pascal*		100	**84**
Turbo POWER TOOLS *for Turbo Pascal*		100	**84**
View Manager *for C or Pascal*		275	**209**
with Source Code	*Special Price*	295	**239**

borland products

REFLEX *Data Base System*	*New*	99	**75**
Turbo DATABASE TOOLBOX		55	**38**
Turbo EDITOR TOOLBOX		70	**54**
Turbo GAMEWORKS TOOLBOX		70	**54**
Turbo GRAPHIX TOOLBOX		55	**38**
Turbo LIGHTNING		99	**75**
Turbo PASCAL		70	**49**
with 8087 or BCD		110	**77**
with 8087 & BCD		125	**84**
Turbo Prolog Compiler	*New*	100	**CALL**
Turbo TUTOR *for Turbo PASCAL*		35	**28**

C-86 Compiler	*See Computer Innovations Section*	395	**289**
Datalight C Compiler *with large memory model*	*New*	99	**79**
DeSmet C Compiler *w/Source Debugger*		159	**145**
Eco-C *Complete Development System* by Ecosoft		125	**CALL**
Lattice C *with Free SecretDisk*	*See Feature Page*	500	**299**
Let's C Compiler by Mark Williams		75	**69**
with csd Source Level Debugger		150	**129**
MWC-86 by Mark Williams	*Special Price*	495	**289**
Microsoft C Compiler *with source debugger.*	*New version*	495	**CALL**
Wizard C by Wizard Systems	*Includes lint*	450	**369**

c interpreters

C-terp by Gimpel Software	*Specify compiler interface*	300	**239**
Instant C by Rational Systems		500	**379**
Interactive C by IMPACC Associates	*New*	249	**219**
Run/C from Lifeboat		150	**99**
Run/C Professional from Lifeboat		250	**189**

c utilities

Please refer to the following sections for additional products: Blaise, Computer Innovations, Lattice, Microsoft, Phoenix, Polytron, SoftCraft and Xenix System V.

Application Programmer's Toolkit by Shaw American		395	**339**
BasicC Library by C Source		175	**135**
C Essentials by Essential Software	*New*	100	**85**
C-lib by vance info systems		195	**125**
C Power Packs by Software Horizons		CALL	**CALL**
c-tree by FairCom	*Includes source*	395	**329**
C Utility Library by Essential Software		185	**139**
C Windows by Syscom		100	**89**
C Wings by Syscom		50	**45**
dbVISTA *Single-User DBMS* by Raima		195	**159**
with Source Code		495	**429**
dbVISTA *Multi-User DBMS* by Raima		495	**429**
with Source Code		990	**849**
Entelekon C Function Library		130	**115**
Entelekon C Windows		130	**115**
Entelekon Superfonts for C		50	**45**
Entelekon Combination Package		200	**175**
Essential Graphics by Essential Software.	*No royalties*	250	**219**
Flash-up Windows by Software Bottling of NY		75	**69**
GraphiC by Scientific Endeavors	*Mono version 2.2*	280	**219**
GraphiC by Scientific Endeavors	*Color version 3.0*	350	**299**
The Greenleaf Functions by Greenleaf Software		185	**135**
Greenleaf Comm Library by Greenleaf Software		185	**135**
The HAMMER by OES Systems		195	**175**
H.E.L.P. by Everest Solutions		90	**79**
MetaWINDOWS by Metagraphics		185	**139**
MetaWINDOWS/Plus by Metagraphics	*New*	235	**199**
Multi-Halo by Media Cybernetics	*Royalties*	300	**219**
On-line Help from Opt-Tech Data		149	**119**
PANEL by Roundhill	*Library Source Available*	295	**229**
PC Lint by Gimpel Software		139	**109**
Scientific Subroutine Library for C by Peerless		175	**139**
Vitamin C by Creative Programming		150	**139**
VC Screen *Forms Designer*	*Requires Vitamin C*	99	**85**
Zview by Data Management Consultants		245	**199**

cobol language

Micro Focus COBOL Workbench		4000	**3599**
Micro Focus Level II COBOL		CALL	**CALL**
COMATH		200	**169**
FORMS-2		300	**269**
Level II Animator		1200	**995**
Level II FILESHARE		800	**659**
Level II SOURCEWRITER		2000	**1689**
Micro Focus Micro/SPF		175	**159**
Micro Focus Professional COBOL		3000	**2395**
Microsoft COBOL	*See Microsoft Section*	700	**495**
Realia COBOL	*New*	995	**795**
RM/COBOL by Ryan-McFarland		950	**675**
RM/COBOL 8X *ANSI 85 COBOL* by Ryan-McFarland	*New*	1250	**995**

computer innovations products

C-86 Optimizing Compiler		395	**289**
C to dBase		150	**139**
CI Probe *Source Level Debugger*		225	**199**
CI ROM Pack *for C-86*		195	**149**
Introducing C	*C Interpreter*	125	**105**

fortran language

ACS Time Series by Alpha Computer Service		495	**429**
Btrieve *ISAM File Mgr*	*See SoftCraft Section*	-	-
For-Winds by Alpha Computer Service		90	**79**
Forlib-Plus by Alpha Computer Service		70	**55**
Microsoft Fortran	*Links with Microsoft C*	350	**215**
MORE FORTRAN by Peerless Engineering	*New*	125	**99**
Multi-Halo by Media Cybernetics	*Royalties*	300	**219**
PANEL *Screen Designer* by Roundhill		295	**229**
PC Fortran Tools by Stat Com Systems	*New*	179	**159**
PolyFortran Tools by Polytron		179	**139**
RM/Fortran by Ryan-McFarland		595	**395**
Scientific Subroutine Library by Peerless		175	**139**
Scientific Subroutine Package by Alpha Computer		295	**259**
The Statistician by Alpha Computer Service		295	**259**
Strings & Things by Alpha Computer Service		70	**55**

(6/86)

lattice products

Refer to our feature page preceding this spread for more information.

		LIST	OURS
Lattice C Compiler		500	299
with Library Source Code		900	549
C Cross Reference Generator		50	39
with Source Code	New	200	159
C-Food Smorgasbord *Function Library*		150	99
with Source Code		300	195
C-Sprite *Source Level Debugger*		175	139
Curses *Screen Manager*		125	99
with Source Code		250	199
dBC *dBase File Manager for C*		250	199
with Source Code		500	395
LMK *Make Facility*		195	149
LSE *Screen Editor*	New	125	99
RPG II Compiler		750	595
SecretDisk *File Security*		60	49
SideTalk *Resident Communications*		120	95
Text Mgmt Utilities *(GREP/DIFF/ED/WC/Extract/Build)*		120	95
TopView Toolbasket *Function Library*		250	199
with Source Code		500	395
Z-80 C Cross Compiler	New	500	395
with Library Source Code		1000	789

microsoft products

Microsoft BASIC Interpreter *for Xenix*		350	279
Microsoft C Compiler *with source debugger.*	New version	495	CALL
Microsoft COBOL Compiler		700	495
Microsoft COBOL Compiler *for Xenix*		995	795
Microsoft COBOL Tools	with COBOL Source Debugger	350	209
Microsoft COBOL Tools *for Xenix*		450	359
Microsoft Fortran Compiler	Links with Microsoft C	350	209
Microsoft Fortran Compiler *for Xenix*		495	389
Microsoft LISP	New Common Lisp	250	189
Microsoft Macro Assembler *with utilities*		150	99
Microsoft Mouse *Bus Version*	New	175	149
Microsoft Mouse *Serial Version*	New	195	159
Microsoft muMath	Includes muSIMP	300	195
Microsoft Pascal Compiler	Links with Microsoft C	300	195
Microsoft Pascal Compiler *for Xenix*		495	389
Microsoft Sort		195	149
Microsoft QuickBASIC Compiler		99	79
Microsoft Windows		99	74

modula-2 language

MODULA-2/86 Compiler by Logitech		89	65
with 8087		129	105
with 512K		189	149
MODULA-2 Editor by Logitech		59	49
MODULA-2 Runtime Debugger by Logitech		69	59
MODULA-2 Source Package by Logitech		179	155
MODULA-2 Utilities Package by Logitech		49	45

other products

Dan Bricklin's Demo Program by Software Garden	New	75	65
FASTBACK *Backup Utility* by 5th Generation Systems	New	179	159
Interactive EASYFLOW by Haventree Software	New	150	129
Janus/ADA C Pack by R&R Software		95	89
Janus/ADA D Pack by R&R Software		900	699
PC/Forth by Laboratory Microsystems		150	119
PC/Forth+ by Laboratory Microsystems		250	209

phoenix products

Authorized Dealer
Springtime Sale!

Pasm86 *Macro Assembler*		295	179
Pfantasy Pac	*Pfinish,Pfix+,Plink+,Pmaker,Pmate,Ptel*	1295	895
Pfinish *Performance Analyzer*		395	229
Pfix-86 Plus *Symbolic Debugger*		395	229
PforCe *C Function Library*	New	475	CALL
Plink-86 *Overlay Linker*		395	229
Plink-86 Plus *Overlay Linker*		495	359
Pmaker *Program Development Manager*		195	139
Pmate *Macro Text Editor*		225	139
Pre-C *Lint Utility*		395	229
Ptel *Binary File Transfer Program*		195	129

polytron products

Polytron C Library I		99	79
Polytron C Beautifier	New	49	45
PolyFORTRAN Tools I		179	139
PolyLibrarian *Library Manager*		99	79
PolyLibrarian II *Library Manager*		149	119
PolyMake *UNIX-like Make Facility*		99	79
PolyOverlay *Overlay Optimizer*		99	79
Polytron PowerCom *Communications*	New	179	139
PolyWindows Developer Kit	New	199	149
PolyWindows Products	*All varieties*	CALL	CALL
PolyXREF *Cross Reference Utility*	*Complete system*	219	179
PolyXREF	*Support for one language only*	129	109
PVCS *Polytron Version Control System*		395	359
PVMFM *Polytron Virtual Memory File Manager*		199	149

softcraft products

Btrieve *ISAM File Mgr with no Royalties*	New version	250	195
Btrieve/N *for Networks*		595	465
Rtrieve *Report Generator for Xtrieve*		85	79
Rtrieve/N *Report Generator for Xtrieve/N*		175	159
Xtrieve *Query Utility for Btrieve*		195	169
Xtrieve/N *Query Utility for Btrieve/N*		395	299
OPT-Tech Sort *Works with Btrieve Files*	New version	149	119

text editors

Brief *from Solution Systems*		195	CALL
Epsilon by Lugaru	New version	195	165
FirsTime for Turbo by Spruce Technology		75	69
KEDIT by Mansfield Software Group	Like Xedit	125	109
SPF/PC by Command Technology Corp		195	165
Vedit by CompuView		150	115
Vedit Plus by CompuView		125	180
XTC *Text Editor* by Wendin	Includes source	99	84

turbo pascal utilities

Please refer to the following sections for additional products:
Blaise, Borland and SoftCraft.

ALICE by Software Channels	New Pascal Interpreter	95	85
FirsTime for Turbo by Spruce Technology		75	69
Flash-up Windows by Software Bottling of NY		75	69
Multi-Halo *Graphics* by Media Cybernetics	Royalties	250	CALL
On-line Help from Opt-Tech Data		149	119
Screen Sculptor by Software Bottling of NY		125	95
Turbo EXTENDER by TurboPower Software	New	85	69
Turbo Professional by Sunny Hill Software		70	49
TurboRef by Gracon Services		50	45
TurboPower Utilities by TurboPower Software		95	84
TurboWINDOW by MetaGraphics		80	69
XTC *Text Editor* by Wendin		99	84

video training tapes

These video cassette training tapes are from the Information Factory and are an excellent alternative to expensive classroom training. Specify Beta or VHS. Price includes one student manual. Call for more information.

Computer Literacy	New	400	CALL
Local Area Networks	New	350	CALL
Programmer's Introduction to C	New	500	CALL

wendin products

Operating System Toolbox	Build your own OS	99	84
PCUNIX *Operating System*		99	84
PCVMS *Operating System*	Similar to VAX/VMS	99	84
XTC *Text Editor*	Includes Pascal source code	99	84

xenix system v by sco

Xenix Development System	Specify XT or AT	595	529
Xenix Operating System	Specify XT or AT	595	529
Xenix Text Processing Package	Specify XT or AT	195	179
Complete Xenix System	Specify XT or AT	1295	1099

xenix languages and utilities

APL*PLUS/UNX System by STSC	For AT Xenix	995	795
Btrieve *ISAM File Manager* by SoftCraft		595	465
c-tree by FairCom	Includes C source code	395	329
Informix by SCO		995	839
Lyrix by SCO		595	489
Microsoft Languages	See Microsoft Section	-	-
Networks for XENIX by SCO	New	CALL	CALL
PANEL *Screen Designer* by Roundhill	For AT Xenix	595	539
SCO Professional by SCO	Complete Lotus clone	795	695

We are open until 5 p.m. Pacific Time, (8 p.m. Eastern).
Purchase Orders are accepted from qualified accounts at no extra charge.
Visa and MasterCard are accepted with no surcharge applied.
Please include card expiration date when ordering by mail.
Account is charged when shipped.

VISA

MasterCard

CALL TOLL FREE

800-336-1166
U.S. OHIO 216-877-3781

800-225-1166
CANADA

OUR NO RISK GUARANTEE

If you are not completely satisfied with your purchase you may return it within 30 days. All returned products must meet our standards for being in new, resellable condition including all paperwork and unused registration card. Products including source code are generally excluded by the manufacturer from this guarantee. Please ask for specific details when placing your order.

Prices are subject to change without notice.

programmer's connection

136 SUNNYSIDE ST.
HARTVILLE, OHIO 44632

Circle **no. 129** on reader service card.

DDJ ON LINE

Listing One (Text begins on page 16.)

```
* PERMG for the 68000.  Inversely permute a 256-bit bit
* vector, BLOCK, by table BITPERM.  On call:
*
*       a0 -> BLOCK, a 32-byte area
*       a1 -> BITPERM, a table of byte values in 0..255
*
* On return, a0 -> permuted BLOCK
*            All registers saved
*
* Register usage:
*       a2 -> WORK, a 32-byte temporary work area
*       a3 -> BLOCK
*       d0 =  outer loop counter
*       d1 =  inner loop counter
*       d3 =  longword from BLOCK
*       d4 =  byte from BITPERM
*       d5 =  temporary
*       d6 =  #7, immediate masking value
* Version of 3/22/86.  Executes in 4 ms on 8 MHz system.

        .globl  permg
        .globl  work

permg:
        movem.l d0-d6/a0-a3,-(a7)
        moveq   #7,d0           clear work area
        lea     work,a2
clrloop:
        clr.l   (a2)+
        dbf     d0,clrloop
        lea     work,a2         init registers
        move.l  a0,a3           save block addr for later
        moveq   #7,d0           outer loop control
        moveq   #0,d2           count of bits
        move    d2,d4           need word clear
        move    d2,d5           need word clear
        moveq   #7,d6           masking value
permg1:
        moveq   #31,d1          inner loop control
        move.l  (a0)+,d3        get longword from BLOCK
bitloop:
        btst    d1,d3           check for bit on
        bne     setbit          if on, set BITPERM[d2] bit in WORK
permg2:
        addq    #1,d2           ...else, bump count
        dbf     d1,bitloop      and do for all bits in this word
        dbf     d0,permg1       do for all words of BLOCK

movloop:
        move.l  (a2)+,(a3)+     move WORK to BLOCK
        dbf     d6,movloop      use #7 already in d6
        movem.l (a7)+,d0-d6/a0-a3
        rts                     all done

setbit:
        move.b  (a1,d2),d4      get byte BITPERM[COUNT]
        move    d4,d5           save for reuse
        lsr     #3,d4           index to byte of WORK
        and     d6,d5           compute bit # in that byte
        eor     d6,d5           reverse bit order
        bset    d5,(a2,d4)      set the bit in WORK
        bra     permg2          re-enter main loop

        .end
```

End Listing One

Listing Two

```
* PERMF for the 68000.  Permute a 256-bit bit vector, BLOCK
* by table BITPERM.  On call:
*
*       a0 -> BLOCK, a 32-byte area
*       a1 -> BITPERM, a table of byte values in 0..255
*
* On return, a0 -> permuted BLOCK.
*            All registers saved.
*
* Register usage:
*       a2 -> WORK, a 32-byte temporary work area
*       d0 =  byte from BITPERM, shifted to bit index
*       d1 =  index to byte of BLOCK
*       d2 =  #3, immediate shift value
*       d3 =  identifies bit in WORK to change
*       d4 =  loop counter for BITPERM values
*       d5 =  identifies bit in WORK to change
*       d6 =  #7, immediate masking value
*       d7 =  temporary
* Version of 3/22/86.   Execution time at 8 MHz = 6 ms.

        .globl  permf
        .globl  work

        .text

permf:
        movem.l d0-d7/a0-a2,-(a7)

        moveq   #7,d0           clear work area
        lea     work,a2
clrloop:
        clr.l   (a2)+
        dbf     d0,clrloop
        lea     work,a2
        moveq   #0,d3           init counter to bits in WORK
        move    #255,d4         init BITPERM loop counter
        moveq   #7,d6           masking value
        moveq   #3,d2           shift value
        clr     d0
permloop:
        move.b  (a1)+,d0        get byte from BITPERM
        move    d0,d1           we will need it twice
        lsr.w   d2,d1           compute byte index in BLOCK
        and     d6,d0           save lower 3 bits for bit index
        eor     d6,d0           reverse bit order for btst
        btst    d0,(a0,d1)      is bit on in BLOCK?
        bne     permf2          if so, we must set bit in WORK
permf1:
        addq    #1,d3           else, next bit of WORK
        dbf     d4,permloop     and next byte of BITPERM

movloop:
        move.l  (a2)+,(a0)+     move WORK to BLOCK
        dbf     d6,movloop      use #7 already in d6
        movem.l (a7)+,d0-d7/a0-a2
        rts                     all done

permf2:
        move    d3,d5           if BLOCK bit is on...
        lsr.w   d2,d5           find permuted bit in WORK...
        move    d3,d7           save d3 for counter
        and     d6,d7           save lower 3 bits
        eor     d6,d7           reverse bit order for bset
        bset    d7,(a2,d5)      set desired bit in WORK
        bra     permf1          re-enter main loop

        .end

        bset    d7,(a2,d5)
```

End Listings

BLAST®

WANG · Data General · AT&T · PRIME · HARRIS · apple

IBM · digital · HEWLETT PACKARD · MS-DOS · CP/M · UNIX

PC-MINI-MAINFRAME COMMUNICATIONS SOFTWARE

ANY COMPUTER WITH BLAST CAN TALK TO ANY OTHER COMPUTER WITH BLAST, the universal file transfer utility linking many different computers, operating systems, and networks, via RS 232 serial ports.

NO ADD-ON BOARDS TO BUY! BLAST software uses any asynchronous modems or direct connect for fast, error-free data transfer through noisy lines and PBXs, across LANs, and over satellites or packet switched networks.

THE PERFECT LOW-COST LINK FOR PC's, MINIS, MAINFRAMES Transfer binary or text files, or executable commands. Use BLAST standalone, or built it into your application.

$250/Micros **$495-895**/Minis **$2495**/up Mainframes

COMMUNICATIONS RESEARCH GROUP (800)-24-BLAST

8939 Jefferson Hwy. Baton Rouge, LA 70809 (504)-923-0888

Circle **no. 272** on reader service card.

AVAILABLE

BACK ISSUES

1982	1983	1984	1985	1986
#68—June	#77—March	#87—Jan.	#99—Jan.	#111-Jan.
#69—July	#78—April	#88—Feb.	#104—June	#112—Feb.
#70—Aug.	#79—May	#89—March	#108—Oct.	#113—March
#71—Sept.	#80—June	#90—April	#109—Nov.	#114—April
#72—Oct.	#81—July	#91—May	#110—Dec.	
#73—Nov.	#82—Aug.	#92—June		
	#83—Sept.	#93—July		
	#84—Oct.	#94—Aug.		
	#85—Nov.	#95—Sept.		
	#86—Dec.	#96—Oct.		
		#97—Nov.		

TO ORDER: send $5 for 1 issue, $4.50 each for 2-5, $4 each for 6 or more to: Dr. Dobb's Journal, 501 Galveston Drive, Redwood City, CA 94063

Name

Address

City State Zip 3116

Write-Hand Man

"Almost a Sidekick for CP/M"
Ted Silveira—Computer Currents, Aug. 27, 1985

"WHM is ingenious and works as intended"
Jerry Pournelle, BYTE Magazine, Sept. 1985 (c) McGraw-Hill

Now available for CP/M 2.2, CP/M 3.0 and ZRDOS!

The convenience of *Sidekick* on your CP/M machine! Trigger **Write-Hand-Man** with a single keystroke and a window pops open to run desk accessories. Exit **Write-Hand-Man** and both the screen and program are restored. Use with any CP/M program and most any CP/M machine. Takes only 5K of memory.

FEATURES

- Notepad for quick notes
- Appointment calendar
- HEX calculator
- File and Directory viewer
- Quick access phonebook
- 14 digit decimal calculator

BONUS

Add applications written by you or others! No other *"Sidekick"* lets you add applications. Dump screens, setup printers, communicate with other computers, display the date and time. Let your imagination run wild!

$49.95 (California residents add tax), shipping included. COD add $2. Sorry, no credit cards or purchase orders. 30 day guarantee. Formats: 8 inch IBM, Northstar and most 5 inch (please specify).

Write-Hand-Man only works with CP/M 2.2, ZRDOS and CP/M 3.0 (please specify). Simple terminal configuration required. Not available for TurboDOS. Compatible with keyboard extenders, hard disks, and other accessories.

Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
415-493-3735

Trademarks: **Write-Hand-Man** — **Poor Person Software**, CP/M—Digital Research, *Sidekick*—Borland International

Circle **no. 169** on reader service card.

Another in a series of productivity notes on MS-DOS™ software from UniPress.

DATA MANAGEMENT SYSTEM

PHACT™

Subject: Low-cost, powerful multi-user Data Management System with keyed B+ tree Isam file manager, report writer and query facility. Runs on the IBM-PC.™

PHACT was designed specifically for the programmer. PHACT provides a one-product solution for accessing and manipulating records, generating reports, and selecting data through a powerful query facility.

Features:

- *Supports fixed and variable length records (1-9999 bytes).*
- *Up to 9 alternate indices can be utilized.*
- *Record locking for multiple simultaneous updates. Handles Networks!*
- *Full or partial key record access.*
- *Automatic key compression to save disk space and improve performance.*
- *Databases are password protected and may be opened for shared or exclusive use.*
- *Can be linked with these C libraries: Lattice,™ Microsoft, C86, and Aztec.™*
- *PHACT-report provides an easy to use command language for formatting reports from existing PHACT databases.*
- *PHACT-query uses a SEQUEL-like relational query language for selecting data.*
- *PHACT Forms Generator coming soon!*
- *Excellent product to integrate into your own packages. Call for terms on distribution rights for source purchases.*

Price:

PHACT Data Management System —

	Binary	Source
Single-user system	$ 655	$2085
Multi-user system	1220	4985

Available for UNIX™ and VMS™ too!
Mastercard/Visa accepted

***For our Free Catalog** and more information on these and other MS-DOS software products, call or write:*
UniPress Software Inc.
2025 Lincoln Highway
Edison, NJ 08817
Telephone: 201-985-8000
*Order Desk: **800-222-0550 (Outside NJ)***
Telex: 709418
European Distributor
Modulator SA
Switzerland, Telephone: 41 31 59 22 22
Telex: 911859
Japanese Distributor
D.I.T. Inc.
Minato-Ku, Tokyo
Telephone: 03 586-6580

PHACT is a trademark of PHACT Associates; IBM-PC, International Business Machines Corp.; Lattice is a trademark of Lattice, Inc.; MS-DOS is a trademark of Microsoft; Mark Williams is a trademark of Mark Williams; Aztec is a trademark of Aztec.

UniPress Software
Your Leading Source for UNIX™ Software.

Circle **no. 77** on reader service card.

Work Smart with These Powerful C Utilities

Get more value from your C system. Boost program quality and slash development time with these professional utilities for leading C-compiler systems.

C Utility Library ~~$185~~ **$155**
Over 300 C subroutines
C and assembler source code and demonstration programs for screen handling, color printing, graphics, DOS disk and file functions, memory management and peripherals control.

C-tree ~~$395~~ **$329**
B-Tree database system
Store, update and retrieve records easily. High-level multi-key ISAM routines and low-level B-Tree functions. Available for MS-DOS, CP/M-86, and CP/M-80. Easily transported. Adaptable for network and multiuser. Includes source.

PHACT ~~$295~~ **$200**
Data Base Record Manager
Includes high-level features found in larger database systems. Available for MS-DOS, CP/M-86 and CP/M-80.

Pre-C ~~$395~~ **$329**
LINT-like source code analyzer
Locates structural and usage errors. Cross-checks multiple files for bad parameter declarations and other interface errors.

Windows for C ~~$195~~ **$165**
Versatile window utility
Supports IBM PC compatible and some non-compatible environments.

PANEL ~~$295~~ **$235**
Screen generating utility
Create custom screens via simple, powerful editing commands. Select colors, sizes and types, edit fields. Includes direct input utility.

HALO ~~$250~~ **$199**
Ultimate C graphics
A comprehensive package of graphics subroutines for C. Supports multiple graphics cards.

PLINK-86 ~~$395~~ **$315**
Overlay linker
Includes linkage editor, overlay management, a library manager and memory mapping. Works with Microsoft and Intel object format.

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)

UNIX is a registered TM of Bell Laboratories. C-tree, TM Faircom, Inc., PHACT TM PHACT ASSOC. Pre-C, PLINK-86, TM PHOENIX. HALO TM Media Cybernetics, Inc., PC-lint TM GIMPLE software. PANEL TM Roundhill Computer Systems, Ltd. WINDOWS FOR C TM Creative Solutions. CP/M TM DRI

Circle **no. 109** on reader service card.

C CHEST

Listing One (Text begins on page 22.)

```
/*
 *                                SORT.C
 *
 *                    Copyright (c) 1986 Allen I. Holub
 *                         All rights reserved.
 */

#include <stdio.h>
#include <ctype.h>
#include <getargs.h>

extern char     *malloc();

/*-----------------------------------------------------------------------
 *      General purpose #defines.
 */

#define MAXBUF          (132 + 1)       /* Maximum input line length +1 */
#define MAXLINEC        1024            /* Maximum number of lines in   */
                                        /* an input file before merge   */
                                        /* files start to be created    */
#define MAXTMP          18              /* The maximum number of temp-  */
                                        /* orary files that will be     */
                                        /* created. Note that fp's are  */
                                        /* needed for stdout, and       */
                                        /* stderr during output         */

#define isnum(c1)       ( isdigit(c1) || (c1) == '-' )

/*-----------------------------------------------------------------------
 *      Variables for getargs. The immediately following variables will
 *      be modified by getargs() according to what flags it finds on the
 *      command line.
 */

static int      Noblanks        =  0    ;       /* Blanks don't count    */
static int      Numeric         =  0    ;       /* Sort numbers by val   */
static int      Primary         =  0    ;       /* Primary sort key      */
static int      Secondary       =  0    ;       /* Secondary sort key    */
static int      Dictorder       =  0    ;       /* Use dictionary order  */
static int      Foldupper       =  0    ;       /* Fold upper case       */
static int      Reverse         =  0    ;       /* Sort in reverse order */
static int      Delim           =  0    ;       /* Field separator       */
static char     *Mdir           =  ""   ;       /* Put temp files here   */
static int      Nodups          =  0    ;       /* Don't print duplicate */
                                                /* lines.                */
ARG     Argtab[] =
{
    /* arg     type      variable    error message string     */

      { 'b' , BOOLEAN,   &Noblanks,   "ignore leading whitespace (Blanks)"},
      { 'd' , BOOLEAN,   &Dictorder,  "sort in Dictionary order"          },
      { 'f' , BOOLEAN,   &Foldupper,  "Fold upper into lower case"        },
      { 'n' , BOOLEAN,   &Numeric,    "sort Numbers by numeric value"     },
      { 'p' , INTEGER,   &Primary,    "use field <num> as Primary key"    },
      { 'r' , BOOLEAN,   &Reverse,    "do a reverse sort"                 },
      { 's' , INTEGER,   &Secondary,  "use field <num> as Secondary key"  },
      { 't' , CHARACTER, &Delim,      "use <C> to separate fields"        },
      { 'T' , STRING,    (int*)&Mdir, "prepend <str> to Temp file names"  },
      { 'u' , BOOLEAN,   &Nodups,     "delete duplicate lines in output"  }
};

#define NUMARGS         (sizeof(Argtab) / sizeof(ARG))

/*-----------------------------------------------------------------------
 *      Global variables. The Lines array is used for the initial
 *      sorting.
 */

static  int     Options;                /* Set by main if any options set */
static  char    *Lines[MAXLINEC];       /* Holds arrays of input lines    */
static  int     Linec;                  /* # of items in Lines            */
static  char    **Argv;                 /* Copies of argv and argc        */
static  int     Argc;

/*-----------------------------------------------------------------------
 *      The heap used in the merge process:
 */

typedef struct
{
        char    string[MAXBUF];         /* One line from the merge file */
        FILE    *file;                  /* Pointer to input file        */
}
HEAP;

HEAP    *Heap[ MAXTMP ];                /* The heap itself              */

/*-----------------------------------------------------------------------*/

#ifndef DEBUG
#define pheap(s,n)
#else

pheap( str, n )
char *str;
```

(continued on page 70)

The Best C Book
A Powerful C Compiler
One Great C Value $39.95

A good C book just isn't complete without a good C compiler to go with it. That's why we give you both. You get a comprehensive 450 page book and a standard, full feature C compiler. It's everything you need to take advantage of this powerful, portable language.

Our book is filled with sample programs. You'll learn how to use pointers to functions with a program that computes the time value of money. A simple data base program illustrates dynamic memory allocation and linked lists. Sample programs are included with the description of all functions.

Our compiler works fast because it makes only one pass through your program. Unlike other C compilers, it doesn't require a separate program to control the compile process. And it won't wear out your disk drives creating intermediate files. One fast pass and you've got an object file that's ready to be linked.

You also get our fast linker and an extensive library of standard C functions. In addition to the portable C functions you get a large number of computer specific functions so you won't have to write them yourself. The library includes an interface to the BDOS and BIOS routines so you can easily add your own special functions when the job demands it.

You can't learn to program in C without a good book and a good compiler. You can buy other C books but they don't include a compiler. You can buy other C compilers but they don't include a book. Either way you spend a lot of bucks and the compiler might not do what the book says it should. With MIX C you don't have to worry. You get both a good book and a good comiler for just a few bucks. And we guarantee that the compiler does what the book says it should.

Language Features

- Data Types: char, short, int, unsigned, long, float, double, void
- Data Classes: auto, extern, static, register
- Typedef, Struct, Union, Bit Fields, Enumerations
- Structure Assignment, Passing/Returning Structures

Functions

abs asm asmx atan atof atoi atol bdos bdosx bios biosx calloc ceil cfree chain character chdir chmod clearerr close clrscrn cmpstr conbuf conc cos cpystr creat cursblk curslin curscol cursrow cursoff curson delete drand exec execl execv exit exitmsg exp fabs fclose fdopen feof ferror fflush fgets fileno filetrap find floor fopen fprintf fputs fread free freopen fscanf fseek ftell fwrite getc getch putch getchar getcseg getdseg getd putd getdate gettime geti puti getkey getmode setmode gets getw heapsiz heaptrap hypot index inp insert iofilter isalnum isalpha isascii iscntrl isdigit islower isprint ispunct isspace isupper itoa keypress left$ len log log10 longjmp lseek malloc alloc mathtrap mid$ mkdir modf movmem open outp peek perror poke poscurs pow printf putc putchar puts putw rand read readattr reach writech readdot writedot realloc rename replace repmem rewind right$ rindex rmdir scanf setbuf setbufsiz setcolor setdate settime setjmp setmem sin sound sprintf sqrt srand sscanf stacksiz str$ strcat strcmp strcpy strlen strncat strncmp strncpy strsave system tolower toupper ungetc ungetch unlink write writechs xmembeg xmemend xmemget xmemput xmovmem _exit

MIX Editor $29.95

When you're programming in a high level language you need a high powered editor. That's why we created the MIX Editor. It's a powerful split screen text processor that works great with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC. It even has fill and justify for English.

You can split the screen horizontally or vertically and edit two files at once. You can move text back and forth between the two windows. You can also create your own macro commands from an assortment of over 100 predefined commands. It comes configured like WordStar but you can customize it to work like other editors or word processors.

The editor works terrific with our C compiler. The MSDOS/PCDOS version has a macro for compiling direct from memory. If your program has an error the editor positions the cursor to the error and displays an error message. You can also run other programs and execute DOS commands. Because the editor works so well with our C compiler we want to make sure you have both. For a limited time we're offering the editor for only $15 when purchased with the C compiler.

ASM Utility $10

The ASM utility allows you to create your own assembly language function libraries. It works with Microsoft's MASM or M80 assemblers. It provides macros for function entry and exit so you don't have to worry about environment details. It also provides a macro for calling C functions from assembly language. Lots of useful assembly language functions are included as examples.

ORDERS ONLY
1-800-523-9520
IN TEXAS
1-800-622-4070

Canadian Distributor
Saraguay Software: 416-923-1500

30 Day Money Back Guarantee **NOT COPY PROTECTED**

Editor $ _____ (29.95)
C $ _____ (39.95)
C & Editor $ _____ (54.95)
ASM Utility $ _____ (10.00)
TX Residents $ _____ (6.125% sales tax)
Shipping $ _____ (see below)
Total $ _____

☐ Check ☐ Money Order ☐ MC/Visa
_____ Exp _____

Shipping Charges: (No charge for ASM Utility)
USA: $5/Order Canada: $10/Order
Overseas: $10/Editor • $20/C • $30/C & Editor

☐ PCDOS/MSDOS (2.0 or later)
☐ IBM PC Single Side
☐ IBM PC Double Side
☐ Tandy 2000
☐ 8 Inch
☐ Other _____
☐ CPM 80 (2.2 or later/Z80)
☐ 8 Inch
☐ Kaypro II
☐ Kaypro 4
☐ Apple (Z80)
☐ Osborne I SD
☐ Osborne I DD
☐ Morrow MD II
☐ Other _____

Name _____
Street _____
City _____
State _____ Zip _____
Country _____
Phone _____

MIX software
2116 E. Arapaho
Suite 363
Richardson, TX 75081
(214) 783-6001
Ask about our volume discounts. D

CPM is a trademark of Digital Research. MSDOS is a trademark of Microsoft. PCDOS is a trademark of IBM. WORDSTAR is trademark of Micro Pro.

Vitamin C™
A Healthy Supplement For C Programmers

Perfect Windows, Powerful Data Entry, FULLY Integrated, For Effortless Data Entry Windows!

- Complete input formatting
- Unlimited Validation
- Full attribute control
- Field sensitive application help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- "Loop function" allows processing while awaiting input

and much much more!

Designed to increase your productivity and help you produce superior applications in dramatically less time! Includes 100% source, tutorial, reference manual, examples, and sample programs.

Specify Microsoft, Lattice, Computer Innovations, Aztec, or Mark Williams. Ask about Unix.

$149.95

NOW... VCScreen !

Our new interactive screen "painter" actually lets you draw your data entry windows! Define fields, text, boxes & borders. Move them around. Change attributes. Then the touch of a button generates C source code calls to the Vitamin C routines! Now, your screen designs will be faster and more pleasing when they are created with VCScreen!

Requires Vitamin C library above. For IBM & compatibles.

$99.95

For Orders Or Information,

(214)245-6090

Creative Programming Consultants
Box 112097 Carrollton, TX 75011-2097
Include $3 ground, $6 air, $15 overnight shipping, $25 if outside USA. Texans add 6% tax. All funds must be in U.S. dollars drawn on a U.S. bank.

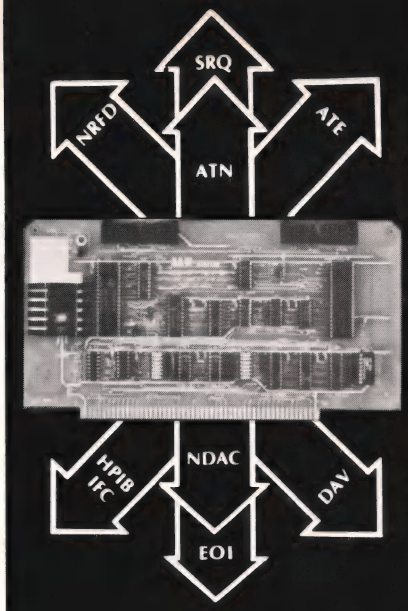
C CHEST

Listing One **(Listing continued, text begins on page 22.)**

```
 97 {
 98         int     i;
 99
100         printf("+-------------------------\n");
101         printf("| %s, heap is:\n", str);
102         for(i = 0; i < n ; i++ )
103         {
104                 printf("|%02d: %s", i, *(Heap[i]->string) ?
105                                         Heap[i]->string  : "(null)\n" );
106         }
107         printf("+-------------------------\n");
108 }
109
110 #endif
111
112 /*-------------------------------------------------------------------------*/
113
114 int     dedupe(argc, argv)
115 int     argc;
116 char    **argv;
117 {
118         /*      Compress an argv-like array of pointers to strings so that
119          *      adjacent duplicate lines are eliminated. Return the
120          *      argument count after the compression.
121          */
122
123         register int    i       ;
124         int             nargc   ;
125         char            **dp    ;
126
127         nargc = 1;
128
129         dp = &argv[1];
130
131         for ( i=1  ;  i < argc ;  i++ )
132         {
133                 if( strcmp(argv[i-1], argv[i]) != 0 )
134                 {
135                         *dp++ = argv[i];
136                         nargc++;
137                 }
138         }
139
140         return( nargc );
141 }
142
143 /*-------------------------------------------------------------------------*/
144
145 static char     *skip_field(n, str)
146 int             n;
147 char            *str;
148 {
149         /*      Skip over n fields. The delimiter is in the global
150          *      variable Delim. Return a pointer to either the character
151          *      to the right of the delimiter, or to the '\0'.
152          */
153
154         while( n > 0   &&   *str )
155         {
156                 if( *str++ == Delim)
157                         --n;
158         }
159
160         return(str);
161 }
162
163 /*-------------------------------------------------------------------------
164  *              Comparison functions needed for sorting.
165  *
166  *      ssort() will call either argvcmp or qcmp, passing them pointers
167  *      to linev entries. qcmp() calls two workhorse functions, qcmp1()
168  *      and qcmp2(). The workhorse functions will also be called by the
169  *      reheap() subroutine used to manipulate merge files.
170  */
171
172 static int      argvcmp( s1p, s2p )
173 char            **s1p, **s2p;
174 {
175         return( strcmp( *s1p, *s2p ) );
176 }
177
178 /*-------------------------------------------------------------------------*/
179
180 static int      qcmp( str1p, str2p )
181 char            **str1p;
182 char            **str2p;
183 {
184         /*      Takes care of all the sorting of fields, calling qcmp1
185          *      to do the actual comparisons. Assuming here that
186          *      Secondary won't be set unless Primary is set too.
187          */
188
189         return( qcmp1( *str1p, *str2p ) );
190 }
191
```

```
192 /*------------------------------------------------------------------------*/
193
194 static int      qcmp1( str1, str2 )
195 char            *str1, *str2;
196 {
197         /*      Workhorse comparison function. Takes care of sorting
198          *      fields. If a primary sort field is specified then
199          *      qcmp1() skips to that field and calls qcmp2 to do the
200          *      actual comparison. If the primary fields are equal, then
201          *      the secondary fields are compared in the same way.
202          */
203
204         int     rval;
205
206         if( !Primary )
207                 return qcmp2( str1, str2 );
208         else
209         {
210                 rval = qcmp2(   skip_field(Primary - 1, str1),
211                                 skip_field(Primary - 1, str2)   );
212
213                 if( !rval && Secondary )
214                 {
215                         /* The two primary keys are equal, search the   */
216                         /* secondary keys if one is specified           */
217
218                         rval = qcmp2(   skip_field(Secondary - 1, str1),
219                                         skip_field(Secondary - 1, str2) );
220                 }
221
222                 return rval;
223         }
224 }
225
226 /*------------------------------------------------------------------------*/
227
228 static int      qcmp2( str1, str2 )
229 char            *str1;
230 char            *str2;
231 {
232         /*      Workhorse comparison function. Deals with all command line
233          *      options except fields. Returns
234          *
235          *              0       if      str1 == str2
236          *              positive if     str1  > str2
237          *              negative if     str1  < str2
238          *
239          *      This is a general purpose (and therefore relatively slow)
240          *      routine. Use strcmp() if you need a fast compare.
241          *      Comparison stops on reaching end of string or on encountering
242          *      a Delim character (if one exists). So make sure Delim is
243          *      set to '\0' if you're not sorting by fields.
244          */
245
246         register unsigned int   c1, c2;
247
248         if( Noblanks )
249         {
250                 /*
251                  *      Skip past leading whitespace in both strings
252                  */
253
254                 while( isspace(*str1) )
255                         str1++;
256
257                 while( isspace(*str2) )
258                         str2++;
259         }
260
261         do
262         {
263                 if( Numeric && isnum(*str1) && isnum(*str2) )
264                 {
265                         /* Add 0xff to the two numeric values so that
266                          * c1 and c2 can't be confused with a Delim
267                          * character later on.
268                          */
269
270                         c1 = stoi( &str1 ) + 0xff ;
271                         c2 = stoi( &str2 ) + 0xff ;
272
273                         if( c1 == c2 )
274                                 continue;
275                         else
276                                 break;
277                 }
278
279                 c1 = *str1++;
280                 c2 = *str2++;
281
282                 if(Dictorder)
283                 {
284                         /*      Skip past any non-alphanumeric or blank
285                          *      characters.
286                          */
287
288                         while( c1 && !isalnum(c1) )
289                                 c1 = *str1++ ;
290
291                         while( c2 && !isalnum(c2) )
292                                 c2 = *str2++ ;
293                 }
```

(continued on next page)



THE 488+3

IEEE 488 TO S-100 INTERFACE

S-100 ⟷ 488

- *Controls IEEE 488 (HP1B) Instruments with an S-100 computer*
- *Acts as controller or device*
- *Basic and assembly language drivers supplied*
- *Meets IEEE 696 specification*
- *Industrial quality burned in and tested up to 125K bytes/sec under software control 3 parallel ports (8255-5)*
- *$375*

D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

Circle **no. 265** on reader service card.

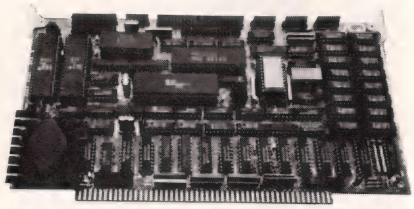
Products With Expandability

A two user Slave card based on Hitachi's Z80 compatible high speed, 10MHz super microprocessor.

Priced at **$495.00***

Features Include. . .

- 4-10 MHz Z80 Compatible HD64180
- 1/2 Megabyte Nonbanked Memory
- 2 Asynchronous Serial Ports To 38.4
- 1 High Speed Synchronous Port
- All Transfers Via 1.6 MHz DMA!!!
- Unique Expansion Port Offers;
  2 Additional Serial Ports or . . .
  2 Parallel Ports or . . .
  Real Time Clock With Battery Backup



The industry's fastest 8-bit Master CPU card with features superior to most 16-bit cards.

Priced at **$495.00***

Each Master Features . . .

- 4-10 MHz Z80 Compatible HD64180
- 1/2 Megabyte Nonbanked Memory
- 2 Asynchronous Serial Ports To 38.4
- 1 High Speed Synchronous Serial Port
- 4 Bi-directional Parallel Ports
- TurboDOS**, ZSYSTEMS**, CP/M**, & OASIS** Operating Systems
- FDC Simultaneously Controls 8", 5¼", & 3½" Drives
- SASI/SCSI Interface
- Optional High Speed Hard Disk/File Access Tape Backup and True ETHERNET Controller

**Prices apply to 6 MHz, 64KB versions and are good for a limited time only on purchases of ten or more. For less than ten, please call.*

***Trademarks: TurboDOS - Software 2000; ZSYSTEMS - Echelon; CP/M - Digital Research; OASIS - THEOS Software*

iCD INTELLIGENT COMPUTER DESIGNS CORP.

23151 Verdugo Drive, Suite 113
Laguna Hills, CA 92653
(714) 581-7500

Circle **no. 278** on reader service card.

C CHEST

Listing One **(Listing continued, text begins on page 22.)**

```
294
295
296                 if(Foldupper)
297                 {
298                         /* Map c1 and c2 to upper case  */
299
300                         c1 = toupper( c1 );
301                         c2 = toupper( c2 );
302                 }
303
304                 /* Keep processing while the characters are the same
305                  * unless we've reached end of string or a delimiter.
306                  */
307         }
308         while( (c1==c2)  &&  c1  &&  c1 != Delim );
309
310         if( Delim )                     /* If we're sorting on a field  */
311         {                               /* and we've found a delimiter  */
312                 if(c1 == Delim)         /* then map the delimiter to a  */
313                         c1 = 0;         /* zero for purposes of         */
314                                         /* comparison.                  */
315                 if(c2 == Delim)
316                         c2 = 0;
317         }
318
319         return( Reverse ? (c2 - c1) : (c1 - c2) );
320 }
321
322 /*------------------------------------------------------------------------*/
323
324 FILE    *nextfile()
325 {
326         /*      Return a FILE pointer for the next input file or NULL
327          *      if no more input files exist (ie. all of the files
328          *      on the command line have been processed) or a file
329          *      can't be opened. In this last case print an error message.
330          *      If Argc == 1 the first time we're called, then standard
331          *      input is returned (the first time only , NULL is returned
332          *      on subsequent calls).
333          */
334
335         FILE            *fp;
336         static  int     first_time = 1 ;
337
338         if( first_time )
339         {
340                 first_time = 0;
341                 if( Argc == 1 )
342                         return stdin;
343         }
344
345         if( Argc-- > 1 )
346         {
347                 if( !(fp = fopen(*++Argv, "r")) )
348                         fprintf(stderr, "sort: can't open %s for read\n",
349                                                                 *Argv );
350                 return fp;
351         }
352
353         return NULL;
354 }
355
356 /*------------------------------------------------------------------------*/
357
358 gtext()
359 {
360         /*      Get text from the appropriate input source and put
361          *      the lines into Linev, updating Linec. Return non-zero
362          *      if more input remains. Note that non-zero will
363          *      be returned if there are exactly MAXLINEC lines in
364          *      the input, even though there isn't any more actual
365          *      input available. If malloc can't get space for the line,
366          *      we'll remember the line in buf and return 1.
367          */
368
369         register int    c;
370         static FILE     *fp             =  0;
371         static char     buf[ MAXBUF ]   = {0};          /* Input buffer */
372         int             maxcount;
373         char            **lv;
374
375         if( !fp )                       /* This is only true the first  */
376                 fp = nextfile();        /* time we're called.           */
377
378         lv    = Lines ;
379         Linec = 0     ;
380
381         for( maxcount = MAXLINEC;  --maxcount >= 0 ;)
382         {
383                 if( !*buf )
384                         while( fgets(buf, MAXBUF, fp) == NULL )
385                                 if( !(fp = nextfile()) )
386                                         return( 0 );        /* No more input */
387
```

(continued on page 81)

Product Information

Free!

Dr. Dobb's Journal of Software Tools

June 1986 #116 Expiration Date: September 30, 1986

Name _____ Title _____

Company _____ Phone _____

Address _____

City / State / Zip _____

Please circle one letter in each category: S

I. My work is performed:
A. for in-house use only.
B. for other companies.
C. for end users/retailers.
D. in none of the above areas.

II. My primary job function:
A. Software Project Mgmt/Spvr
B. Hardware Project Mgmt/Spvr
C. Computer Consultant
D. Corporate Management
E. Other

III. My company department performs:
A. software development.
B. computer system integration.
C. computer manufacturing.
D. computer consulting.
E. computer research.
F. none of the above.

IV. This inquiry is for:
A. a purchase within 1 month.
B. a purchase within 1 to 6 months.
C. product information only.

V. Corporate Purchase Authority:
A. Final Decision-maker
B. Approve/Recommend
C. No Influence

VI. Personal Computer Users at my Jobsite:
A. 10,000 or more
B. 500 to 9,999
C. 100 to 499
D. 10 to 99
E. less than 10

VII. On average, I advise others about computers:
A. more than once per day.
B. once per day.
C. once per week.
D. less than once per week.

VIII. In my job function, I:
A. design software and/or write code.
B. design software.
C. write code.
D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001	002	003	004	005	006	007	008	009
010	011	012	013	014	015	016	017	018
019	020	021	022	023	024	025	026	027
028	029	030	031	032	033	034	035	036
037	038	039	040	041	042	043	044	045
046	047	048	049	050	051	052	053	054
055	056	057	058	059	060	061	062	063
064	065	066	067	068	069	070	071	072
073	074	075	076	077	078	079	080	081
082	083	084	085	086	087	088	089	090
091	092	093	094	095	096	097	098	099
100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162
163	164	165	166	167	168	169	170	171
172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198
199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216
217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234
235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261
262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288
289	290	291	292	293	294	295	296	297
298	299	999						

Circle 999 to start a 12 month subscription at the price of $29.97

Postage Paid!

Thank You!
Dr. Dobb's greatly appreciates your responses to questions I through VIII.

Product Information

Free!

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
P.O. Box 2157
Clinton, Iowa 52735-2157

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
P.O. Box 2157
Clinton, Iowa 52735-2157

Product Information

Free!

Thank You!
Dr. Dobb's greatly appreciates your responses to questions I through VIII.

Postage Paid!

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001	002	003	004	005	006	007	008	009
010	011	012	013	014	015	016	017	018
019	020	021	022	023	024	025	026	027
028	029	030	031	032	033	034	035	036
037	038	039	040	041	042	043	044	045
046	047	048	049	050	051	052	053	054
055	056	057	058	059	060	061	062	063
064	065	066	067	068	069	070	071	072
073	074	075	076	077	078	079	080	081
082	083	084	085	086	087	088	089	090
091	092	093	094	095	096	097	098	099
100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162
163	164	165	166	167	168	169	170	171
172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198
199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216
217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234
235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261
262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288
289	290	291	292	293	294	295	296	297
298	299	999						

Circle 999 to start a 12 month subscription at the price of $29.97

Dr. Dobb's Journal of Software Tools

June 1986 #116 **Expiration Date: September 30, 1986**

Name _____ Title _____

Company _____ Phone _____

Address _____

City/State/Zip _____

Please circle one letter in each category: 5

I. My work is performed:
A. for in-house use only.
B. for other companies.
C. for end users/retailers.
D. in none of the above areas.

II. My primary job function:
A. Software Project Mgmt/Spvr
B. Hardware Project Mgmt/Spvr
C. Computer Consultant
D. Corporate Management
E. Other

III. My company department performs:
A. software development.
B. computer system integration.
C. computer manufacturing.
D. computer consulting.
E. computer research.
F. none of the above.

IV. This inquiry is for:
A. a purchase within 1 month.
B. a purchase within 1 to 6 months.
C. product information only.

V. Corporate Purchase Authority:
A. Final Decision-maker
B. Approve/Recommend
C. No Influence

VI. Personal Computer Users at my Jobsite:
A. 10,000 or more
B. 500 to 9,999
C. 100 to 499
D. 10 to 99
E. less than 10

VII. On average, I advise others about computers:
A. more than once per day.
B. once per day.
C. once per week.
D. less than once per week.

VIII. In my job function, I:
A. design software and/or write code.
B. design software.
C. write code.
D. don't design software or write code.

Product Information

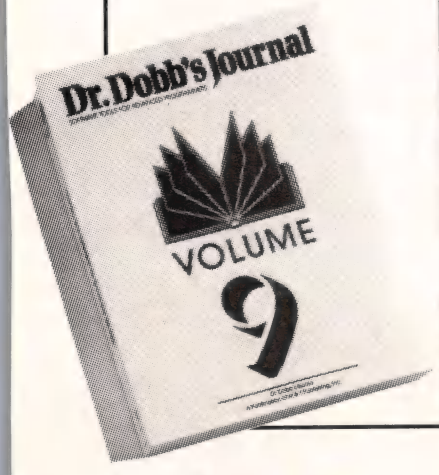
Free!

Dr. Dobb's Catalog

Continuing the tradition of providing its readers with useful, powerful software tools,* Dr. Dobb's Journal of Software Tools *presents this collection of programming tools in the form of books and software on disk.

Dr. Dobb's Bound Volume 9

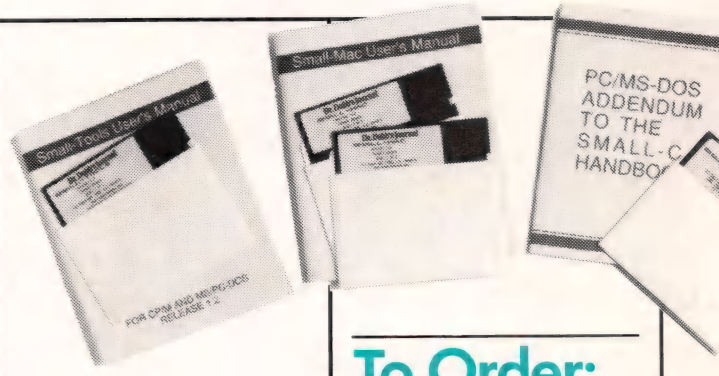
This ninth volume in the ***Dr. Dobb's Bound Volume*** series contains the complete editorial contents of ***Dr. Dobb's Journal of Software Tools*** for 1984: over 1000 pages of text and code. To commemorate the release of Volume 9, we are offering a discount on the full set of Bound Volumes. This set includes every issue of *DDJ* from 1976 through 1984. If you order the set now, you will receive a $40 discount.



DDJ Listings on Disk

You've demanded it, and we're delivering.

Dr. Dobb's Journal of Software Tools now offers a listings service. The first disk has just been released, containing most of the program listings that appeared in the magazine from January through April 1986. This accompaniment to the magazine is available in various formats.



Also Inside

- **Dr. Dobb's Complete Toolbox of C**
- **A Unix-like Shell for MS DOS** and
- **A Unix-like Utility Package for MS DOS** by Allen Holub
- **Dr. Dobb's Z80 Toolbook** by David Cortesi

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

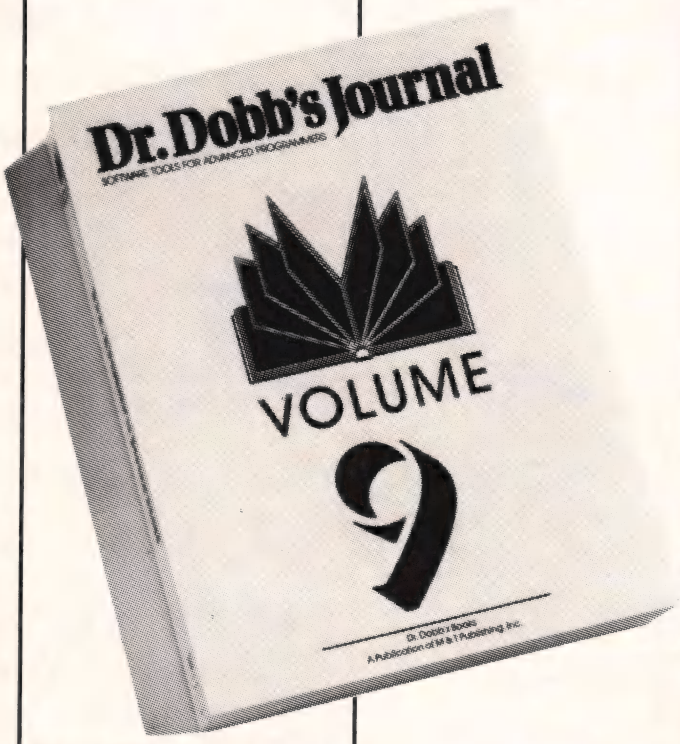
M&T BOOKS

Dr. Dobb's Bound Volumes

Programs by the Pound

Announcing: Dr. Dobb's Bound Volume 9

Over 1000 pages of listings and text. The entire 1984 editorial contents of *Dr. Dobb's Journal of Software Tools.*



Bound Volume 9: 1984
Item #020B

Shaping things to come. In 1984 new Editor-in-chief Mike Swaine brought his interests in advanced technology to *Dr. Dobb's Journal.* We presented the concepts behind Prolog and published an expert system for weather prediction. We learned Modula-2, and taught Forth to talk to a 68000, to MS DOS, and to the people of China. We examined a new language implementation called Turbo Pascal and extended implementations of C with a preprocessor, a library, Tony Skjellum's tricks, and Allen Holub's Grep. We published two powerful encryption systems, telecommunications protocols, floating-point benchmark results, and an issue devoted to the internals of Unix. And Ray Duncan, Bob Blum, and Dave Cortesi were on hand with their fascinating columns.

Bound Volume 1: 1976
Item #013

The working notes of a technological revolution. Programmers from Defense laboratory systems analysts to kitchen-table entrepreneurs worked for the intrinsic rewards to put development software on the brand-new invention, the microcomputer. Before there was an Apple, *Dr. Dobb's Journal of Tiny Basic Calisthenics and Orthodontia* (subtitle: Running Light without Overbyte) was founded to put a programming language on the machines, and became both chronicler and instrument of the revolution. In this first-year volume: Tiny Basic, the first word on CP/M, notes on building an IMSAI, floating-point and timer routines.

Bound Volume 2: 1977
Item #014

Running light without overbyte. By year two, *Dr. Dobb's* formula was concocted: tough questions and serious technical issues handled with enthusiasm, wit, and scant reverence for the accepted answers. Source code. Tools for programmers. Respect for tight programming. *Dr. Dobb's Journal* readers shared insights on warping the Intel 8080 into a computer CPU, and *Dr. Dobb's* published a complete operating system for the chip. A motley crop of computers and software products were popping up, and *Dr. Dobb's* investigated: the Heath H-8, the KIM-1, the Alpha Micro, MITS Basic, Poly Basic, and Lawrence Livermore Labs Basic. *Dr. Dobb's* introduced Pilot for microcomputers and published tips on doing string handling, high-speed I/O, and turtle graphics in limited memory.

Bound Volume 3: 1978
Item #015

The roots of Silicon Valley growth. In 1978 Steve Wozniak and other programmers were publishing in *Dr. Dobb's Journal* code that would help them grow multimillion-dollar computer companies. The proposed S-100 bus standard was hashed out in *Dr. Dobb's* pages. *Dr. Dobb's* contributors began to speak more in terms of technique than of specific implementations as the industry began to diversify. Languages covered in depth included SAM76, Pilot, Pascal, and Lisp.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001
and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

Dr. Dobb's Bound Volumes

Programs by the Pound

Bound Volume 4: 1979

Item #016

In the midst of the Gold Rush. Three years before IBM would release its PC, a thriving, rough-and-tumble personal computer industry existed. Fortunes had been made and lost, the effective power of the machine multiplied a hundredfold. By 1979 some stability had even emerged; one could speak of the processors that had proven longevity as microcomputer CPUs: the 8080, the Z80, the 6800, and the 6502. *Dr. Dobb's Journal* focused on the best ways to use these processors, with algorithms, tips, and code for 8- to 16-bit conversion, pseudo-random number generation, micro-to-mainframe connections, telecommunications, and networking. And lots of useful code.

Bound Volume 5: 1980

Item #017

The preeminence of CP/M and the rise of C. More than any other magazine, *Dr. Dobb's Journal* was responsible for the spread of CP/M and C on microcomputers. Both of those movements began in 1980. *Dr. Dobb's* all-CP/M issue, including Gary Kildall's history of CP/M, sold out within weeks of publication. This was the year of Ron Cain's original Small C compiler, of a CP/M-oriented C interpreter, CP/M-to-UCSD Pascal file conversion techniques, and of a greater concern in *Dr. Dobb's* with software portability.

Bound Volume 6: 1981

Item #018

The first of Forth. 1981 saw *Dr. Dobb's* first all-Forth issue (now sold out), along with an emphasis on CP/M, C, telecommunications, and new languages. David Cortesi began "Dr. Dobb's Clinic," one of the magazine's most popular features. Highlights included information on PCNET, the Conference Tree, the Electronic Phone Book, Tiny Basic for the 6809, writing your own compiler, and a systems programming language.

Bound Volume 7: 1982

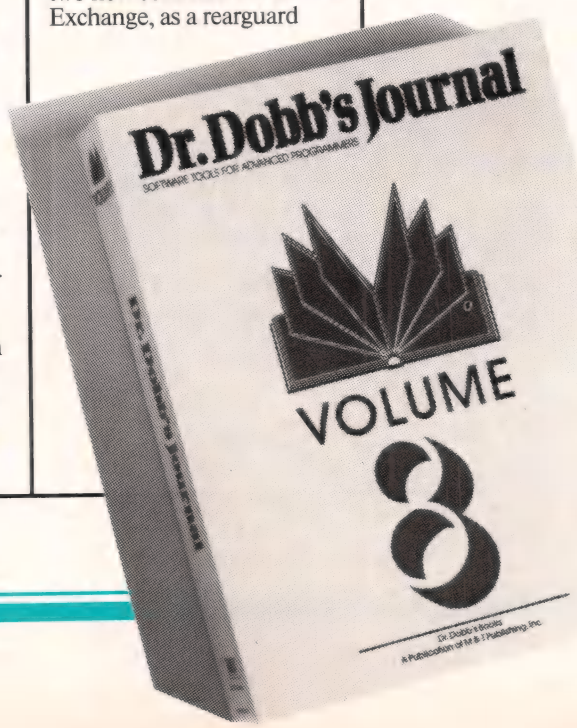
Item #019

Legitimacy. By 1982 IBM had become a player in the personal computer game and was changing the rules. New microprocessors arrived, the first designed specifically to serve as personal computer CPUs. In *Dr. Dobb's Journal* Dave Cortesi published the first serious comparison of MS DOS and CP/M-86. *Dr. Dobb's* started two new columns: the CP/M Exchange, as a rearguard maneuver to ensure that good tools for CP/M programmers would continue to be developed and circulated, and the 16-Bit Software Toolbox to investigate the 8088/86 and other new microprocessors. We published code for the 68000 and Z8000 processors, and looked ahead, in a provocative essay, to fifth-generation computers.

Bound Volume 8: 1983

Item #020

Power tools. Personal computers were proving themselves to be true professional software development tools by 1983, the year in which Jim Hendrix completed his "canonical" version of Small C in *Dr. Dobb's Journal*. *Dr. Dobb's* published more 68000 and 8088 code, and as the memory limitations relaxed, the magazine's commitment to tight code let it shoehorn impossibly large systems into memory. Small C was just one of the major software products published in their entirety in *Dr. Dobb's* pages that year; there were Ed Ream's RED screen editor and a version of the Ada language called Augusta.



Buy the complete set and save 15%

If you buy all nine volumes, covering the entire editorial content of ***Dr. Dobb's Journal of Software Tools*** from the first issue in 1976 through 1984, you pay just $225. That's a 15% discount and over $40 off the combined individual prices. To order the complete set of Bound Volumes 1 through 9, ask for item #020C.

Item #013	$27.75	Vol. 1
Item #014	$27.75	Vol. 2
Item #015	$27.75	Vol. 3
Item #016	$27.75	Vol. 4
Item #017	$27.75	Vol. 5
Item #018	$27.75	Vol. 6
Item #019	$30.75	Vol. 7
Item #020	$31.75	Vol. 8
Item #020B	$35.75	Vol. 9

All 9 volumes
Item #020C $225.00

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001
and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T
PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

A collection of powerful tools for C software developers, including books, software on disk, and reference materials from the publisher of Dr. Dobb's Journal of Software Tools

Dr. Dobb's Complete C Toolbox

From M&T Publishing and Brady Communications...

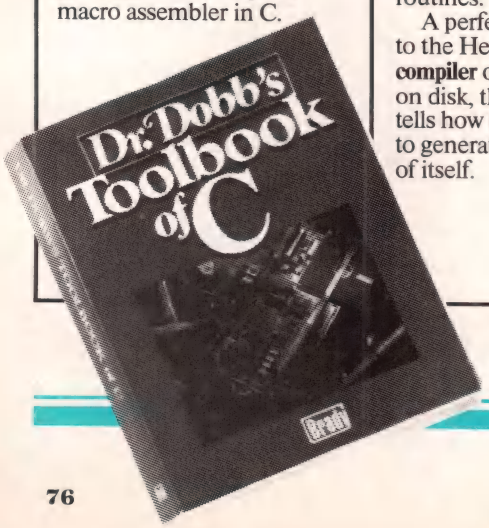
Dr. Dobb's Toolbook of C

Item #005

The **Toolbook** contains over 700 pages of C material, including articles by such C experts as Kernighan and Ritchie, Cain and Hendrix, Skjellum and Holub. The level is sophisticated and pragmatic, appropriate for professional C programmers.

The most valuable part of the **Toolbook** to many will be the hundreds of pages of useful C source code, including a complete compiler, an assembler, and text-processing utilities. The accompanying text explains, in the programmers' own words, why they did what they did.

Dr. Dobb's Journal of Software Tools introduced a generation of personal computer programmers to the C programming language, and all the best C articles and code published in *Dr. Dobb's* over the years is included and updated in the **Toolbook,** including Ron Cain's original Small C article and articles from sold-out issues. But the **Toolbook** also includes material never before published, including Jim Hendrix's complete macro assembler in C.



Dr. Dobb's offers the **Toolbook** in a **special hard-bound edition** for just $29.95. You'll find:
Jim Hendrix's famous **Small C Compiler** and **New Library for Small C** (both also available on disk),
NEW: Hendrix's **Small Mac: An Assembler for Small C** and **Small Tools: Programs for Text Processing** (both also available on disk),
All of Tony Skjellum's **C Programmer's Notebook** columns distilled by Tony into one thought-provoking chapter.

Also from M&T Publishing and Brady Communications...

The Small C Handbook

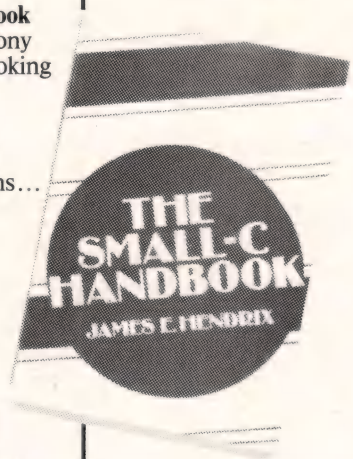
Item #006 or #006A

Jim Hendrix's **Small-C Handbook** is the reference book on his Small-C compiler. In addition to describing the operation of the compiler, the book contains complete source listings to the compiler and its library of arithmetic and logical routines.

A perfect companion to the Hendrix **Small-C compiler** offered by *Dr. Dobb's* on disk, the **Handbook** even tells how to use the compiler to generate a new version of itself.

While both the **Handbook** and the **Toolbook** provide documentation for the Small-C compiler, the **Handbook** contains a more detailed discussion and is available with an addendum for the MS/PC DOS version.

The **Handbook,** Item #006, is just $17.95. Jim Hendrix has ported the compiler to the MS/PC DOS environment since the **Handbook**



was printed, and the **Handbook** plus his **MS/PC DOS Handbook Addendum,** Item #006A, is $22.95.

Dr. Dobb's C Software Tools on Disk

To complement the **Toolbook,** *Dr. Dobb's* also offers the following programs on disk. Full C source code and documentation is included. Except where indicated, both CP/M and MS/PC DOS versions are available.

Small-C Compiler

Item #007

Jim Hendrix's Small-C Compiler is the most popular piece of software ever published in *Dr. Dobb's* 10-year history. Like a home-study course in compiler design, the **Small-C Compiler** and **Small-C Handbook** provide everything you need but the computer for learning how compilers are constructed, and for learning C at its most fundamental level.

While both the **Handbook** and the **Toolbook** provide documentation for the Small-C compiler, the **Handbook** contains a more detailed discussion and is available with an addendum for the MS/PC DOS version. The **MS/PC DOS Small-C Handbook Addendum** is recommended in addition to the **Handbook** for MS DOS or PC DOS users.

The **Small-C Compiler** is available for $19.95 in either the CP/M or the MS/PC DOS version.

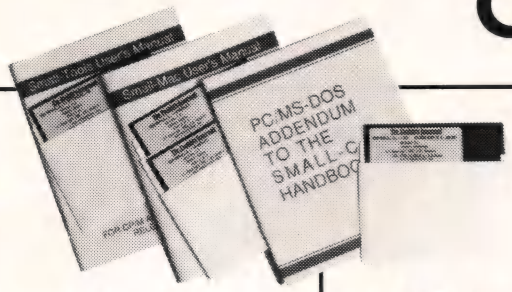
To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or
CALL TOLL-FREE
1-800-528-6050 EXT. 4001
and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

Dr. Dobb's Complete C Toolbox



Small-Mac: An Assembler for Small-C

Item #012A

Small-Mac is an assembler designed to stress simplicity, portability, adaptability, and educational value. The package features a simplified macro facility, C language expression operators, object file visibility, descriptive error messages, and an externally-defined machine instruction table. You get the macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and a utility to dump relocatable files.

Small-Mac is available with documentation for $29.95. For CP/M systems only.

Small-Tools: Programs for Text Processing

Item #010A

A package of programs performing specific, modular operations on text files, including: editing; formatting; sorting; merging; listing; printing; searching; changing; transliterating; copying; concatenating; encrypting and decrypting; replacing spaces with tabs and tabs with spaces; counting characters, words, or lines; and selecting printer fonts. Supplied in source code form so you can select and adapt these tools to your own purposes.

Small-Tools is available with documentation for $29.95. For CP/M or MS/PC DOS systems.

Special Packages — 20% Off

Now for almost 20% off the combined individual product prices, you can order a complete set of *Dr. Dobb's* C programming tools for your CP/M or MS/PC DOS system.

C Package for CP/M

Item #005A

Ordered individually, these items would cost about $120. If you order the CP/M C package, you'll get ***Dr. Dobb's Toolbook,*** the **Small-C Handbook,** the **Small-C Compiler** on disk, the **Small-Mac** assembler on disk with documentation in the **Small-Mac Manual,** the **Small-Tools** text-processing programs on disk with documentation in the **Small-Tools Manual,** all for just $99.95.

C Package for MS/PC DOS

Item #005B

These items would cost over $100 if purchased individually. If you order the MS/PC DOS C package, you'll get ***Dr. Dobb's Toolbook,*** the **Small-C Handbook** with the **MS/PC DOS Addendum,** the **Small-C Compiler** on disk, the **Small-Tools** text-processing programs on disk with documentation in the **Small-Tools Manual,** all for just $82.95.

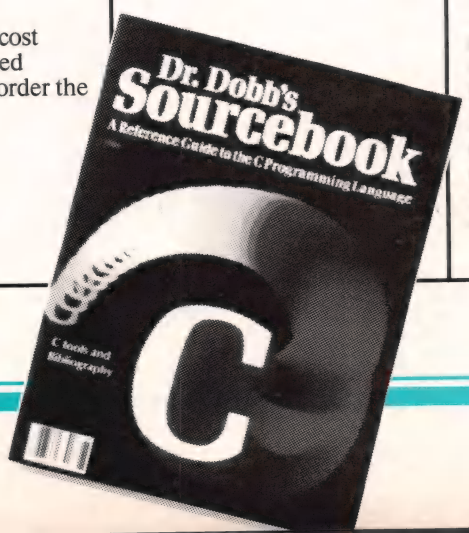
Dr. Dobb's Sourcebook: A Reference Guide to the C Programming Language

Item #004

Products and services for C programmers are appearing at so rapid a rate that it's all but impossible to keep up with them. *Dr. Dobb's* presents this handy guide to the who, what, when, where, and why of C. A comprehensive reference manual to new information, products, and services, the **Sourcebook** contains:

- A bibliography of over 300 articles and books on the C language;
- A descriptive list of products for C programmers: compilers, editors, interpreters, and utilities;
- A list of C-related services: classes, seminars, and on-line services.

The **Sourcebook** is just $7.95.



Dr. Dobb's Sourcebook: A Reference Guide to the C Programming Language
Item #004 $ 7.95

Dr. Dobb's Toolbook for C
Item #005 $29.95

The Small-C Handbook
Item #006 $17.95

The Small-C Handbook and MS/PC-DOS Addendum
Item #006A $22.95

Small-C Compiler disk
Item #007 $19.95

Small Tools: Programs for Text Processing disk
Item #010A $29.95

Small Mac: An Assembler for Small-C disk (For CP/M only.)
Item #012A $29.95

CP/M C Package
Item #005A $99.95

MS/PC DOS C Package
Item #005B $82.95

For CP/M disks, please specify one of the following formats: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001
and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

A UNIX-like Shell and Utility Package for MS-DOS

Only $29.95 each! Both for only $50— save over 15%!

Includes complete source code and documentation.

A UNIX-like Shell for MS-DOS and A UNIX-like Utility Package for MS-DOS
by *Dr. Dobb's* C-Chest columnist, ALLEN HOLUB

The Shell

An MS-DOS implementation of the most often used parts of the UNIX C shell. This package includes an executable version of the shell, along **with complete C source code** and full documentation.
Supported features are:
Editing Command line editing with the cursors is supported. The line is visible as you edit it.
Aliases Can be used to change the names of commands or as very fast memory resident batch files.
History The ability to execute a previous command again. The command can be edited before being executed.
Shell Variables Macros that can be used on the command line.
Nested batch files A batch file can call another batch file like a subroutine. Control is passed to the second file and then back to the first one when the second file is finished. DOS doesn't have this capability.

UNIX-like syntax Slash (/) used as a directory separator, minus (−) as a switch designator. A 2048 byte command line is supported. Command line wild card expansion. Multiple commands on a line.

The shell also supports redirection of standard input, standard output, and standard error.

This version corrects several bugs found in the original version printed in *Dr. Dobb's Journal,* December 1985 through March 1986 issues. It runs on any MS-DOS computer.

Add additional features to the Shell with

/util

A UNIX-like Utility Package for MS-DOS

This collection of utility programs for MS-DOS includes updates of the highly acclaimed Dr. Dobb's articles *Grep: A UNIX-Like Generalized Regular Expression Processor, Ls* from *Dr. Dobb's* C Chest column, and *Getargs* from DDJ's C Chest.

Source code is included and all programs (and most of the utility subroutines) are fully documented in a UNIX-style manual. You'll find executable versions of:

cat A file concatenation and viewing program
cp A file copy utility
date Prints the current time and date
du Prints amount of space available and used on a disk
echo Echoes its arguments to standard output
grep Searches for a pattern defined by a regular expression
Ls Gets a sorted directory
mkdir Creates a directory
mv Renames a file or directory. Moves files to another directory.
p Prints a file, one page at a time
pause Prints a message and waits for a response
printenv Prints all the environment variables
rm Deletes one or more files
rmdir Deletes one or more directories
sub Text substitution utility. Replaces all matches of a regular expression with another string.

Order The Shell and /util together for only $50! SAVE OVER 15%!

Item #160 The Shell	$29.95
Item #161 /util	$29.95
Item #162 Shell/util Package	$50.00

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

Finally!
You've Asked For It For Years and Here It Is!

Dr. Dobb's Listings On Disk

Dr. Dobb's Listings On Disk

Dr. Dobb's Journal of Software Tools has always provided its readers with valuable code. Now, as a useful adjunct to the magazine, DDJ offers the additional value and convenience of selected listings on disk! ***Dr. Dobb's Listings #186*** is a collection of listings from Dr. Dobb's January 1986 issue, through the April 1986 issue. In this first of three ***Dr. Dobb's Listings*** disks for 1986 you'll find listings from the following DDJ articles, among others.

From Issue #111 January 1986

***A Simple OS for Real-time Applications; 68000 assembly language techniques for an operating system kernel* by DDJ editor Nick Turner

***Exec calls and Fortran; a technique allowing execution of a user or system task from a user program* from DDJ's 16-Bit Software Toolbox, by Robert Sypek

From Issue #112 February 1986

***Fast Integer Powers for Pascal; An implementation of the fastest-known algorithm for the computation of integer powers* by Dennis E. Hamilton

***Data Abstraction with Modula-2; Construction of a priority queue in Modula-2* by Bill Walker and Stephen Alexander

***Learning Ada on a Micro; A draw poker program in Ada* by Do-While Jones

***Fast IBM PC graphics routines* from DDJ's 16-Bit Software Toolbox, by Dan Rollins

***32-bit Square Roots; An 8086 assembly-language routine for 32-bit square roots* by Michael Barr

From Issue #113 March 1986

***Recursive Bose-Nelson Sort; An alternative to Joe Celko's September 1985 sort routine* by R.J. Wissbaum

***A Variable-Metric Minimizer; A C program for minimizing arbitrary functions* by Joe Marasco

***Concurrency and Turbo Pascal; An approach to implementing coroutines in Pascal* by Ernest Bergmann

***Speeding MS DOS Disk Access; Programs to test disk-access speed* by Greg Weissman

***Square Roots on the NS32000; Comparable square root routines in C and assembly language for National Semiconductor's 32000 family* by Richard Campbell

From Issue #114 April 1986

***Boca Raton Inference Engine; Lisp, Prolog, and Expert-2 techniques and code* by Robert Brown

Item #170 $14.95
Dr. Dobb's Listings #186
Please specify MS/DOS, Macintosh, or CP/M. For CP/M disks, please specify one of the following formats: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

David E. Cortesi
longtime Dr. Dobb's columnist and author of *Inside CP/M*
brings you —

Dr. Dobb's Z80 Toolbook

Dr. Dobb's Z80 Toolbook

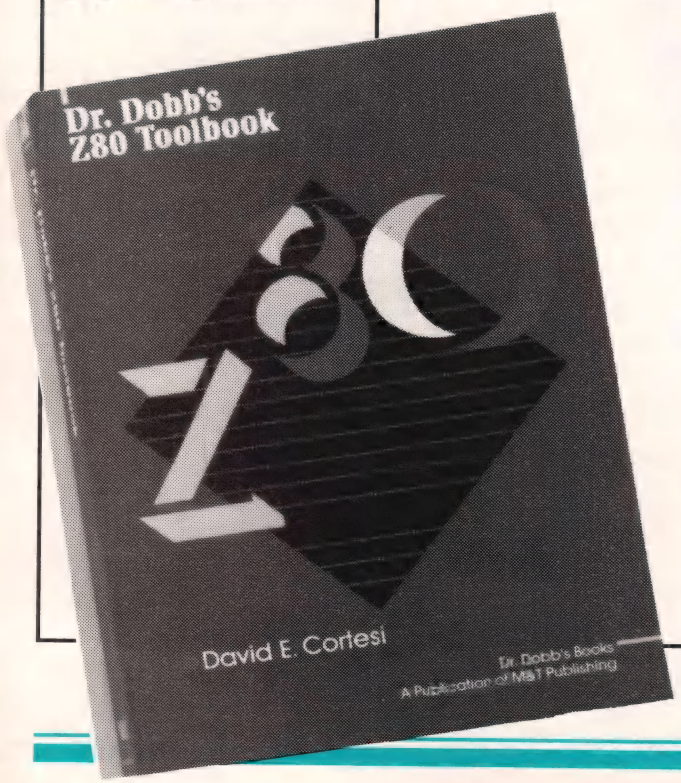
Here's all you need to write your own Z80 assembly language programs for only $25!

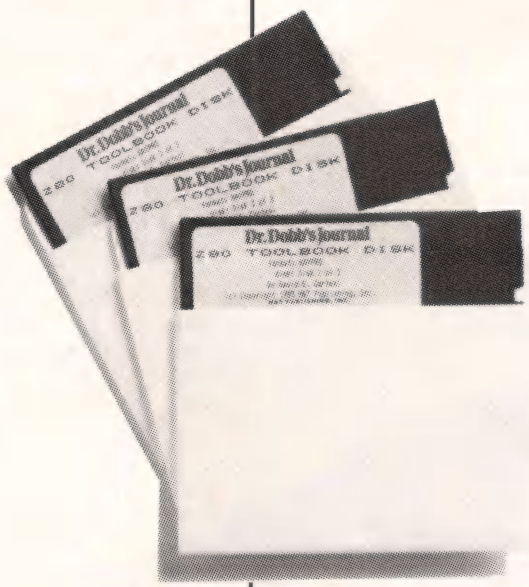
Do you use CP/M? Do you feel as if the only part of the computer industry that has **not** abandoned you is your own Z80 computer? It keeps on working, but when you need programs for it, you have to write them yourself. When you do, you quickly find that while Pascal or BASIC is okay for some things, there is often no substitute for the speed, small size, and flexibility of an assembly language program.

Dr. Dobb's Z80 Toolbook puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

****A method of designing programs and coding them in assembly language.** Cortesi will take you on a walk through the initial specifications, designing an algorithm and writing the code. He demonstrates this method in the construction of several complete, useful programs.

****A complete, integrated toolkit of subroutines** for arithmetic, for string-handling, an for total control of the CP/M file system. They bring the ease and power of a compiler's runtime library to your assembly language work, without a compiler's size and sluggish code. **Best of all, every line of the toolkit's sorce code is there for you to read,** and every module's operation is explained with the clarity and good humor for which Dave Cortesi's writing is known.





Order the Z80 Software on Disk! Save Yourself the Frustration of File Entry

All the software in **Dr. Dobb's Z80 Toolbook**—the programs plus the entire toolkit, both as source code and as object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. (A Z80 microprocessor and a Digital Research International RMAC assembler or equivalent are required.)

Receive *Dr. Dobb's Toolbook for Z80,* along with the software on disk, together for only $40!

Item #022 $25
Dr. Dobb's Toolbook for Z80

Item #022A $40
Dr. Dobb's Toolbook for Z80, together with software on disk. Please specify one of the following formats: 8" SS/SD; Apple; Osborne; Kaypro.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 209

M&T BOOKS

Dr. Dobb's Catalog Order Form

ORDER NOW!

SOLD TO:

Name _____

Street _____ (Please use street address, not P.O. box)

City _____ State _____ Zip _____

Day Phone ( ) _____

For Faster Service on Credit Card Orders

Call Toll Free
1-800-528-6050 Ext. 4001

Please Refer to Item # When Ordering

Or, fill out this postage-paid, self mailing order form and return to: M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063.

For disk orders, please indicate format. Refer to ad for standard format availability for each product.

☐ MS/DOS
☐ CP/M ___Kaypro ___Apple ___Zenith Z-100 DS/DD ___Osborne ___8" SS/SD
☐ Macintosh

Special formats available for additional $10 per product. Please inquire.

	Quantity	Item #	Description	Price Ea.	Total Price
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
				Sub Total	
				Sales Tax	
				Shipping	
				Total Order	

CA residents add applicable sales tax on merchandise total ___%___
(CA residents must add tax to all items EXCEPT *Dr. Dobb's Sourcebook,* Item #004) →

Shipping must be included with order. See rates below. →

☐ Check

Make checks payable to M&T Publishing, Inc.

☐ VISA
☐ MasterCard
☐ American Express

Name on Card _____

Account No. _____

Expiration Date _____

Signature _____

In U.S. For Bound Volumes, add $2.25 per book. Add $8.75 for special C Packages. For other books and disks, add $1.75 per item.

Outside U.S. For Bound Volumes, add $5.25 per book surface mail. Add $18 surface mail for Special C Packages. For other books and disks, add $3.25 per item surface mail. Foreign airmail rates available on request.

Prompt Delivery!
Dealer Inquiries Welcome

M&T BOOKS
3116S

Dr.Dobb's Catalog Order

Please Rush!

Please fold along fold-line and staple or tape closed.

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

First Class Permit No. 790 Redwood City, CA

Postage Will Be Paid By Addressee

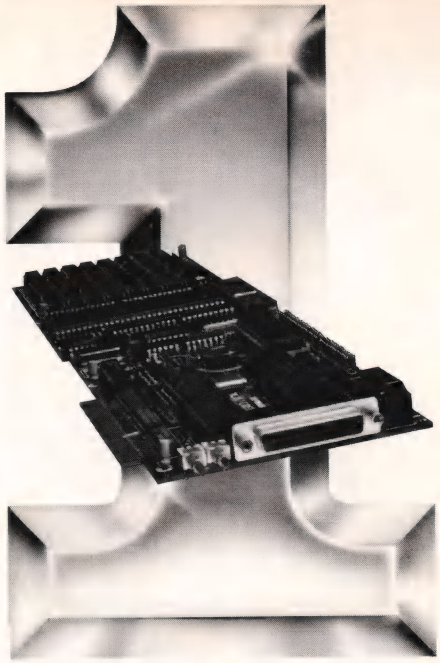
Dr.Dobb's Catalog
501 Galveston Dr.
Redwood City, CA 94063

Please fold along fold-line and staple or tape closed.

Listing One (Listing continued, text begins on page 22.)

```
388                 if( *lv = malloc(strlen(buf) + 1) )
389                 {
390                         strcpy( *lv++, buf );
391                         *buf = '\0';
392                         Linec++;
393                 }
394                 else
395                         return 1;
396         }
397
398         return( maxcount < 0 );  /* Return 1 if there's more input to get */
399 }
400
401 /*-------------------------------------------------------------------------*/
402
403 char    *fname( num )
404 {
405         /* Return a merge file name for the indicated merge pass.
406          */
407
408         static char     name[ 16 ];
409
410         if( num > MAXTMP )
411         {
412                 fprintf(stderr,"sort: input file too large\n" );
413                 exit(1);
414         }
415
416         sprintf(name, "%smerge.%d", Mdir, num );
417         return( name );
418 }
419
420 /*-------------------------------------------------------------------------*/
421
422 outtext( passnum, more_to_go )
423 {
424         /*      Print out all the strings in the Lines array and free all
425          *      the memory that they use. Output is sent to standard
426          *      output if this is pass 1 and there's no more input
427          *      to process, otherwise output is sent to a merge file.
428          */
429
430         register char   **lv;
431         register FILE   *fp;
432
433         if( passnum == 1  &&  !more_to_go )
434                 fp = stdout;
435
436         else if( !(fp = fopen( fname(passnum), "w" )) )
437         {
438                 fprintf(stderr,"Can't open merge file %s for write\n",
439                                                         fname(passnum));
440                 exit(1);
441         }
442
443         for( lv = Lines ; --Linec >= 0; )
444         {
445                 fputs( *lv,   fp );
446                 free ( *lv++     );
447         }
448
449         fclose( fp );
450 }
451
452 /*-------------------------------------------------------------------------*/
453
454
455 open_mergefiles( nfiles )
456 {
457         /*      Open all the merge files and create the heap. "nfiles"
458          *      merge-files exist and the heap will have that many
459          *      elements in it. The heap is unsorted on exit.
460          */
461
462         HEAP    **hp;
463         int     i;
464
465         for( hp = Heap, i = nfiles;   i > 0;  hp++, --i )
466         {
467                 if( ! (*hp = (HEAP *) malloc(sizeof(HEAP))) )
468                 {
469                         fprintf( stderr, "sort: out of memory!" );
470                         exit( 1 );
471                 }
472
473                 if( !( (*hp)->file = fopen( fname(i), "r")  ))
474                 {
475                         fprintf(stderr,"sort: can't open %s for read",
476                                                         fname(i) );
477                         exit( 1 );
478                 }
479
480                 if( !fgets( (*hp)->string, MAXBUF, (*hp)->file ) )
481                 {
```

(continued on next page)



Number One in Performance

68010/68000 Coprocessor for IBM/AT/XT/PC-

8/10/12.5mz No Wait States

$1295.00 Qty. 1

FEATURES

- 1-2 MB RAM (1MB Standard)
- 16K-64K EPROM
- 2-8 Serial Ports
  Async/Sync/Bisync Communications
- Battery-backed Real Time Clock
- Battery-backed 2K-8K RAM
- 2 Parallel Ports
- 68881 Math Coprocessor
- Memory-mapped Dual-port BUS
- 3-9 Users Per Board (3 Standard)
- Up To 16 Boards Per AT/XT/PC
- Can Operate As Standalone Processor

SOFTWARE

- OS9 (Powerful UNIX-like Multi-user OS)
- CPM/68K
- Software selectable OS including concurrent PC DOS/OS-9 or CPM/68K operation
- Support Module for IBM Graphics
- High-speed Local/Global Disk Caching
- Basic, Pascal, Fortran, C, and COBOL

IBM is a registered trademark of International Business Machines
OS-9 is a registered trademark of Microware Systems Corp IBM-XT is a registered trademark of Digital Research Corp
MC68000/MC68010 are registered trademarks of Motorola UNIX is a registered trademark of AT&T

TLM Systems

West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle **no. 173** on reader service card.

Thinking about C?

Stop Thinking-
Start Programming Today!

SPECIAL INTRODUCTORY OFFER!
C' Prime, **Personal Computing and C**,
Plus Apprentice C.
A $169 value only **$99**

NEW FROM MANX AZTEC!

C' Prime ~~$99~~ $79

Never has C been easier to learn. **Manx Aztec** is now offering a complete C system called **C' Prime** at an exceptionally low price. This powerful system includes a Compiler, Linker, Assembler, Editor, Libraries and Object Librarian. **C PRIME** supports a host of third-party software.

C Apprentice ~~$49.95~~ $39.95

Learn C quickly with this complete, easy-to-use C language interpreter. Apprentice C includes a complete one-step compiler that executes with lightning speed, an editor, and a run-time system.

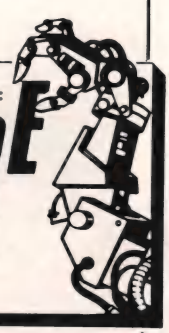
NEW FROM ASHTON-TATE!

Personal Computing and C

A detailed, easy-to-understand guide to C programming prepared especially by Ashton-Tate for use with the new Aztec **C' Prime.** Includes chapters on C programming basics, function libraries, data handling, and advanced features, plus a complete money management demonstration program.

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories, dBase TM Ashton-Tate, Inc., MANX AZTEC C PRIME, Apprentice C TM Manx Software Systems, Inc.

Circle **no. 222** on reader service card.

C CHEST

Listing One **(Listing continued, text begins on page 22.)**

```
482                             fprintf(stderr,"sort: merge file %s is empty",
483                                                             fname(i) );
484                             exit( 1 );
485                     }
486             }
487 }
488
489 /*-------------------------------------------------------------------------*/
490
491 mcmp( hpp1,  hpp2 )
492 HEAP      **hpp1, **hpp2;
493 {
494             /*      Comparison routine for sorting the heap. Is passed
495              *      two pointers to HEAP pointers and compares the
496              *      string fields of these using the same workhorse
497              *      functions used in the initial sorting phase.
498              */
499
500             return  Options ? qcmp1  ((*hpp1)->string, (*hpp2)->string)
501                             : strcmp ((*hpp1)->string, (*hpp2)->string)
502                             ;
503 }
504
505 /*-------------------------------------------------------------------------*/
506
507 reheap( nfiles )
508 {
509             /*      Reheap the Heap, assume that the first element (**Heap)
510              *      is the newly added one.
511              */
512
513             register int    parent, child;
514             HEAP            *tmp;
515
516             for( parent = 0, child = 1; child < nfiles; )
517             {
518                     /*      Find the smaller child. Then if the parent is less
519                      *      than the smaller child, we're done. Otherwise
520                      *      swap the parent and child, and continue the
521                      *      reheaping process with a new parent.
522                      */
523
524                     if( child+1 < nfiles )      /* if child+1 is in the heap */
525                         if( mcmp(&Heap[child], &Heap[child+1]) > 0)
526                             child++;
527
528                     if( mcmp( &Heap[parent], &Heap[child]) <= 0)
529                             break;
530
531                     tmp             = Heap[parent];     /* Exchange             */
532                     Heap[parent] = Heap[child ];
533                     Heap[child ] = tmp;
534
535                     parent = child;
536                     child  = parent << 1;           /*   child = parent * 2 */
537             }
538 }
539
540 /*-------------------------------------------------------------------------*/
541
542 merge( nfiles )
543 int     nfiles;                         /* Number of merge files        */
544 {
545             open_mergefiles( nfiles );
546             ssort( Heap, nfiles, sizeof(Heap[0]), mcmp );
547
548             while( nfiles > 0 )
549             {
550                     pheap( "Merge: top of while loop", nfiles );
551
552                     fputs( (*Heap)->string, stdout );
553
554                     if( !fgets((*Heap)->string, MAXBUF, (*Heap)->file) )
555                     {
556                             /* This input stream is exhausted. Reduce the
557                              * heap size to compensate. Note that Heap+1
558                              * is the same as &Heap[1];
559                              */
560
561                             fclose( (*Heap)->file );
562                             if( --nfiles )
563                                     memcpy(Heap, Heap+1, nfiles * sizeof(HEAP));
564                     }
565
566                     reheap( nfiles );
567             }
568 }
569
570 /*-------------------------------------------------------------------------*/
571
572 adjust_args()
573 {
574             /*      Adjust various default arguments to fix mistakes made
575              *      on the command line. In particular Delim is always 0
576              *      unless either Primary or Secondary was set.
577              *      If a secondary field is specified without a Primary, then
578              *      1 is assumed for the primary. If no Delim is specified
```

```
579         *       then tab (\t) is assumed. "Options" is true if any of
580         *       the options that affect the sort order were specified
581         *       on the command line.
582         */
583
584         if( !(Primary || Secondary) )
585                 Delim = 0;
586         else
587         {
588                 if( !Delim )
589                         Delim = '\t';
590
591                 if( !Primary )
592                         Primary = 1;
593         }
594
595         Options = Noblanks || Numeric || Dictorder || Foldupper
596                                                   || Reverse || Delim;
597 }
598
599 /*-------------------------------------------------------------------------*/
600
601 main(argc, argv)
602 int     argc;
603 char    **argv;
604 {
605         int     numpasses = 0;  /* Number of merge files used           */
606         int     more_input;     /* True if input isn't exhausted        */
607
608         Argc = getargs( argc, argv, Argtab, NUMARGS );
609         Argv = argv;
610         adjust_args();
611
612         do{
613                 more_input = gtext();
614
615                 if( Linec )
616                 {
617                         ssort(Lines, Linec, sizeof(*Lines),
618                                                 Options ? qcmp : argvcmp);
619                         if( Nodups )
620                                 Linec = dedupe(Linec, Lines);
621
622                         outtext( ++numpasses, more_input );
623                 }
624
625         } while(  more_input );
626
627
628         if( numpasses > 1 )             /* merge files were created     */
629         {
630                 fclose( stdin  );       /* Free up default file des-    */
631                 fclose( stdaux );       /* criptors for unused streams  */
632                 fclose( stdprn );       /* so that they can be used for */
633                                         /* merge files.                 */
634                 merge( numpasses );
635
636                 for(; numpasses > 0 ; --numpasses )
637                         unlink( fname(numpasses) );
638         }
639
640         exit(0);
641 }
```

Listing 1 -- stoi.c

End Listing One

Listing Two

```
 1 #include <ctype.h>
 2
 3 int             stoi(instr)
 4 register char   **instr;
 5 {
 6         /*      Convert string to integer updating *instr to point
 7          *      past the number. Return the integer value represented
 8          *      by the string.
 9          */
10
11         register int    num  = 0 ;
12         register char   *str     ;
13         int             sign = 1 ;
14
15         str = *instr;
16
17         if( *str == '-' )
18         {
19                 sign = -1 ;
20                 str++;
21         }
22
23         while( '0' <= *str  &&  *str <= '9' )
24                 num = (num * 10) + (*str++ - '0') ;
25
26         *instr = str;
27         return( num * sign );
28 }
```

End Listings

Number One In Performance

Hard Disk Intelligent VCR Backup for AT/XT/PC

FEATURES

- High speed microprocessor controlled backup (68000)
- Two channel interface
- Built in LAN channel
- Software control of most VCR functions including Fast Forward, Rewind, and auto backup using VCR timer capabilities
- Economical VHS or Beta formats

TLM Systems

West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle **no. 174** on reader service card.

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS/FORTH

Why HS/FORTH? Not for speed alone, although it is twice as fast as other full memory Forths, with near assembly language performance when optimized. Not even because it gives MANY more functions per byte than any other Forth. Not because you can run all DOS commands plus COM and EXE programs from within HS/FORTH. Not because you can single step, trace, decompile & dissassemble. Not for the complete syntax checking 8086/8087/80186 assembler & optimizer. Nor for the fast 9 digit software floating point or lightning 18 digit 8087 math pack. Not for the half megabyte LINEAR address space for quick access arrays. Not for complete music, sound effects & graphics support. Nor the efficient string functions. Not for unrivaled disk flexibility — including traditional Forth screens (sectored or in files) or free format files, all with full screen editors. Not even because I/O is as easy, but far more powerful, than even Basic. Just redirect the character input and/ or output stream anywhere — display, keyboard, printer or com port, file, or even a memory buffer. You could even transfer control of your entire computer to a terminal thousands of miles away with a simple >COM <COM pair. Even though a few of these reasons might be sufficient, the real reason is that we don't avoid the objections to Forth — WE ELIMINATE THEM!

Public domain products may be cheap; but your time isn't. Don't shortchange yourself. Use the best. Use it now!

HS/FORTH, complete system: $395. with "FORTH: A Text & Reference" by Kelly and Spies, Prentice-Hall and "The HS/FORTH Supplement" by Kelly and Callahan

VISA Visa Mastercard MasterCard

HARVARD
SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

Circle no. 132 on reader service card.

LINE GLITCHES

Listing One (Text begins on page 32.)

```
/*
 * short demo program to illustrate Hamming forward error correction
 * code detects and corrects all one bit errors and detects two
 * bit errors in a total transmitted block of 16 bits.
 * eleven bits are message bits, the rest are error checks
 *
 * implementation is oriented toward exposition, not speed or
 * efficiency -- this is not industrial strength code!
 *
 * bit fields not implemented in C/80
 *
 * Joe Marasco, March 1986
 */

#include "fprintf.h"
#define             EOF       -1

#define             B0        1
#define             B1        2
#define             B2        4
#define             B3        8
#define             B4        16
#define             B5        32
#define             B6        64
#define             B7        128
#define             B8        256
#define             B9        512
#define             B10       1024
#define             B11       2048
#define             B12       4096
#define             B13       8192
#define             B14       16384
#define             B15       32768

#define             C1        (B1 + B3 + B5 + B7 + B9 + B11 + B13 + B15)
#define             C2        (B2 + B3 + B6 + B7 + B10 + B11 + B14 + B15)
#define             C3        (B4 + B5 + B6 + B7 + B12 + B13 + B14 + B15)
#define             C4        (B8 + B9 + B10 + B11 + B12 + B13 + B14 + B15)

main()
{
                    int       input[11] ;       /* input message bits        */
register unsigned int         xmit ;            /* transmitted message       */
                    int       recvd[16] ;       /* received message bits*/
register unsigned int         rec ;             /* received message*/
                    int       syndrome ;        /* computed syndrome         */
                    int       recpar    ;       /* parity of rec'd msg       */
                    int       i , ch ;

          for (;;) {

                    printf("input 11 message bits, ^C to quit\n") ;
                    printf("1 2 3 4 5 6 7 8 9 0 1\n") ;

                    i = 0 ;
                    while ( ((ch = getchar()) != EOF) && i<11 )
                              switch(ch) {
                              case '0' :        input[i++] = 0 ;
                                                break ;
                              case '1' :        input[i++] = 1 ;
                                                break ;
                              }
/*
 * anything but a 0 or 1 is ignored
 * check that we have 11 good bits
 */
                    if ( i < 11 ) {
                              printf("not enough valid bits, try again\n") ;
                              continue ;
                    }
/*
 * build the message
 */
                    xmit = 0 ;
                    xmit |= input[0] << 3 ;
                    xmit |= input[1] << 5 ;
                    xmit |= input[2] << 6 ;
                    xmit |= input[3] << 7 ;
                    xmit |= input[4] << 9 ;
                    xmit |= input[5] << 10 ;
                    xmit |= input[6] << 11 ;
                    xmit |= input[7] << 12 ;
                    xmit |= input[8] << 13 ;
                    xmit |= input[9] << 14 ;
                    xmit |= input[10] << 15 ;
/*
 * and the check bits -- even parity
 */
                    xmit |= (parity( C1 & xmit ) << 1) |
                              (parity( C2 & xmit ) << 2) |
                              (parity( C3 & xmit ) << 4) |
                              (parity( C4 & xmit ) << 8)  ;
```

(continued on page 86)

The Best Debuggers. Period.

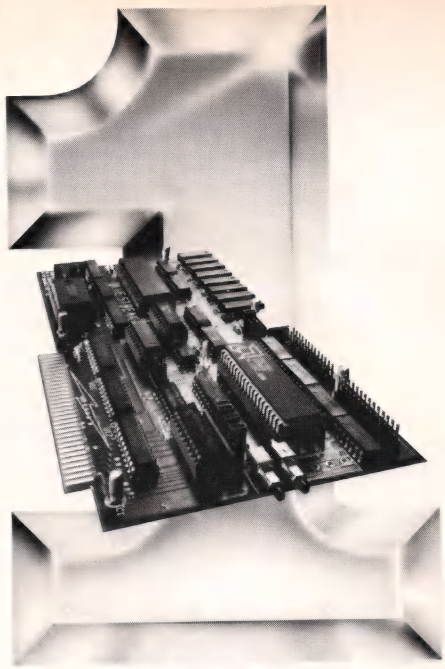
DSD86, The PC-DOS Debugger *69.95*
DSD87, The PC-DOS Debugger with 8087 Support . *99.95*
DSD80, The CP/M Debugger *125.00*

SoftAdvances VISA MasterCard

P.O. Box 49473 • Austin, Texas 78765 • (512) 478-4763

1-800-232-8088

Circle **no. 83** on reader service card.



Number One in Performance

Z80H BLUESTREAK™

IBM/AT/XT/PC- 8mz No Wait States

FEATURES

- 64K-256K RAM
- 2K-8K EPROM/Static Ram
- 2 Serial Ports Async/Sync/Bisync Communications
- Real Time Clock
- Memory-mapped Dual-port BUS
- On-board/Remote Reset NMI capability
- Up To 32 Boards Per AT/XT/PC
- Can Operate As Standalone Processor
- Less Than Full Size Board (will fit other compatables.)

SOFTWARE

- ZP/M tm CP/M Emulation Software (Supports Most CP/M Software)
- Multiuser Capability If Used As A Slave Processor

IBM is a registered trademark of International Business Machines
CPM-80 is a registered trademark of Digital Research Corp

TLM Systems

West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle **no. 175** on reader service card.

Brand New From Peter Norton
A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

only **$50**

Direct from the man who gave you *The Norton Utilities, Inside the IBM PC,* and the *Peter Norton Programmer's Guide.*

THE NORTON EDITOR™

Easily customized, and saved
Split-screen editing
A wonderful condensed/outline display
***Great for* assembler, Pascal *and* C**

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities.* You can ***program your way to glory*** with *The Norton Editor.*"

Peter Norton

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing.

Circle **no. 243** on reader service card.

Listing One (Listing continued, text begins on page 32.)

```
/*
 * and last but not least, make total parity even
 */
                    xmit |= parity( xmit ) ;
/*
 * display it
 */
                    printf("the block sent is                %x \n", xmit ) ;
                    printf("0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5\n") ;
                    for (i=0 ; i<16 ; ++i)
                            printf("%d ", ((xmit>>i) & 01) ) ;
                    printf("\n") ;
                    printf("now input the received block\n") ;
                    printf("0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5\n") ;
/*
 * get the received message
 */
readagain:          i = 0 ;
                    while ( ((ch = getchar()) != EOF) && i<16 )
                            switch(ch) {
                            case '0' :          recvd[i++] = 0 ;
                                                break ;
                            case '1' :          recvd[i++] = 1 ;
                                                break ;
                            }
/*
 * anything but a 0 or 1 is ignored
 * check that we have 16 good bits
 */
                    if ( i < 16 ) {
                            printf("not enough valid bits, try again\n") ;
                            goto readagain ;
                    }
                    for (i=0 , rec=0 ; i<16 ; ++i )
                                    rec |= (recvd[i] << i) ;
                    printf("the block received is            %x \n", rec ) ;
/*
 * compute the syndrome
 */
                    syndrome =   parity( C1 & rec ) |
                                 ( parity( C2 & rec ) << 1 ) |
                                 ( parity( C3 & rec ) << 2 ) |
                                 ( parity( C4 & rec ) << 3 )  ;
/*
 * and the parity bit, which should be zero
 */
                    recpar = parity( rec ) ;
/*
 * decision time
 */
                    if ( syndrome == 0 ) {
                            printf("good message!\n") ;
                            if (recpar) {
                                    printf("with reversed parity bit\n") ;
                                    rec = rec ^ 01 ;
                                    printf("the recovered block is           %x\n",
                                                        rec ) ;
                                    printf("0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5\n") ;
                                    for (i=0 ; i<16 ; ++i)
                                            printf("%d ", ((rec>>i) & 01) ) ;
                                    printf("\n") ;
                            }
                            printf("--------------------------------\n") ;
                    }
                    else {
                            if (!recpar) {
                                    printf("two bit errors, can't fix\n") ;
                                    printf("--------------------------------\n") ;
                            }
                            else {
                                    printf("bad bit in position %d\n",
                                                        syndrome ) ;
                                    rec = rec ^ ( 01 << syndrome ) ;
                                    printf("the recovered block is           %x\n",
                                                                    rec ) ;
                                    printf("0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5\n") ;
                                    for (i=0 ; i<16 ; ++i)
                                            printf("%d ", ((rec>>i) & 01) ) ;
                                    printf("\n--------------------------------\n") ;
                            }
                    }
        }
}
parity( message )
unsigned int message ;
{
/*
 * return 1 if odd parity, 0 if even
 */
        int j , k ;
        for ( j=0 , k=0 ; j<16 ; ++j ) k += ( (message >> j) & 01 ) ;
        return( k & 01 ) ;
}
```

End Listing

Betcha our compiler can beat up your compiler.

Or your money back.

If you program in BASIC, Pascal, Fortran or C, you're using yesterday's technology. We know this statement will start a brawl, but it's true. So, before you start a fight you can't win, take a closer look at the Modula-2 language and the Modula-2 Software Development System (M2SDS) from Interface Technologies. Just compare the features and performance of M2SDS to your system. You'll find a new language and a programming environment that's more flexible, much faster and works on any IBM® PC or 100% compatible with 256 K memory or more.

	M2SDS
COMPILE SPEED (MIN:SEC)	
30 LINES	*0:15.58*
300 LINES	*0:25.48*
EXECUTION SPEED (MIN:SEC)	
SIEVE	*0:13.92*
FIBONACCI	*0:53.49*
30X30 MATRIX (8087)	*0:08.84*
FP OPERATIONS	*0:27.56*
FP OPERATIONS (8087)	*0:01.97*
SYNTAX CHECKING EDITOR	*YES*
MULTIPLE WINDOW EDITING	*YES*
EDITOR FILESIZE LIMIT	*MEMORY SIZE*
COMPILE ERROR CALLS EDITOR	*YES*
LINKER	*YES*
PRODUCES .EXE FILES	*YES*
EXECUTABLE CODE SIZE LIMIT	*DISK SPACE*
DOS ACCESS FROM EDITOR	*YES*
DOS ACCESS FROM PROGRAMS	*YES*
8087 SUPPORT STANDARD	*YES*
COPY-PROTECTED DISK	*NO*
COST WITH 8087 SUPPORT	*$50.88/$80.88*

Source: Software Resources, Inc. Sieve program from BYTE, January 1983. Fibonacci program from Dr. Dobb's Journal, February 1985. Matrix program from BYTE, October, 1982. FP Operations program from BYTE, May 1985. M2SDS with or without 8087 uses 8-byte accuracy. Programs compiled with all checking options on. All tests conducted on a standard IBM-PC/XT with 512K of memory and an 8087 math coprocessor.

*Original or back-up diskette may be sent for trade-in. Diskette will be destroyed immediately upon receipt so that your current compiler license agreement is not violated.

IBM is a registered trademark of International Business Machines Corporation.

Trade In and Trade Up. Just to prove that we're not all brag... we'll send you M2SDS for just $50.88 if you mail us your present compiler or interpreter diskette.* That's $30.00 off the regular price. If within 30 days you're not programming faster than ever, just return the diskette and we'll send you your money back.

Heavyweight Champion SDS-XP. If you're ready to move into light-speed, you need SDS-XP. It has everything M2SDS has with a little "punch" added. Like Extended Libraries, M2MAKE and a Foreign Object Module Importer. SDS-XP offers buyers a stout discount when compared with the cost of buying M2SDS and the additional components as add-ons. For a limited time only, SDS-XP is available for $99.00 with compiler trade-in. That's $150.00 off the advertised price of $249.00.

INTERFACE TECHNOLOGIES

3336 Richmond, Suite 200, Houston, TX 77098

1-800-922-9049

(In Texas, call 713/523-8422) Telex: 322127
In Europe call ITC at: Switzerland 41 (1) 700-3037; Netherlands 31 (20) 106922; U.K. 44 (1) 656-7333

Knock Out Bugs $79.95. Announcing M2DEBUG. A symbolic, compact (only 20K RAM) interactive run-time debugger with features most requested by software programmers and system developers. It comes with a Virtual Resource Overlay Operating Manager (VROOM) that speeds up compile time 50%.

Calling All Compilers. So now that you're wise to the limitations of your system, why not trade it in. You'll soon see that it was smarter to switch than fight. And a pretty safe bet.

Here's my diskette. Rush me:

☐ M2SDS for $50.88 each, plus $7 shipping and handling.

☐ SDS-XP for $99.00 each, plus $7 shipping and handling.

Or, send me:

☐ M2SDS for $80.88 each, plus $7 shipping and handling.

☐ SDS-XP for $249.00 each, plus $7 shipping and handling.

☐ M2DEBUG for $79.95

☐ My check is enclosed.
☐ Apply charges to credit card indicated below:

VISA/MasterCard/American Express (circle one).

Credit Card # _____

Expiration Date _____

Signature _____

Name _____
(please print)

Shipping Address _____

_____

City _____

State/Zip _____

Day Phone _____

Texas residents add 6.125% Sales Tax.
International orders add $30 for shipping/handling.
If paying by check, check or draft must be in U.S. dollars drawn on a U.S. bank.

INTERFACE TECHNOLOGIES CORPORATION
3336 Richmond, Suite 200, Houston, Texas 77098

THE PROGRAMMER'S SHOP

helps save time, money and cuts frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

PC Scheme - by T.I. Interactive LISP which uses lexical scoping, block structure, call by value, and tail-recursive semantics. Compiler, EMACS-like editor, DOS. Can support debugging, graphics, windowing. PC $ 95

AI-Expert System Dev't

Arity System - incorporate with C programs, rule & inheritance	PC	$295
Experteach - Powerful, no limit on memory size. Samples	PC	$399
EXSYS - Improved. Debug, file & external program access.	MS	$339
1st Class - by example, interfaces		$250
Insight 1 - probabilities, fast	MS	$ 79
Insight 2 - dB2, language.	MS	$399
Others: APES ($359), Advisor ($949), ES Construction ($100), ESP ($845), Expert Choice ($449)		

AI-Lisp

		List	Our
GC LISP Interpreter - "Common", rich. Interactive tutorial		$495	Call
GC LISP 286 Developer - 2 to 15 meg RAM, compiler & interp.		$1195	Call
Microsoft MuLisp 85			$199
TLC LISP - "LISP-Machine" - like, all RAM, classes, **compiler.**	MS		$225
TransLISP - learn fast	MS		$ 75
WALTZ LISP - "FRANZ LISP" - like, big nums, debug, CPM-80	MS		$149
Others: IQ LISP ($155), BYSO ($125) UNX LISP ($59), IQC LISP ($269)			

AI-Prolog

ARITY Standard - full, 4 Meg Interpreter - debug, C, ASM	PC	$ 350
COMPILER/Interpreter-EXE	PC	$ 795
With Exp Sys, Screen - KIT	PC	$1250
MicroProlog - enhanced	MS	$ 229
MProlog - Improved, Faster	PC	$ 475
Professional MicroProlog	MS	$ 359
Prolog-86 - Learn Fast, Standard, tutorials, samples	MS	$ 95
TURBO PROLOG by Borland	PC	$ 85
Others: Prolog-I ($365), Prolog-2 ($1795)		

AI-Other

METHODS - SMALLTALK has objects, windows, more	PC	$215
QNIAL - Combines APL with LISP. Source or binary.	PC	$375

FEATURES

Dan Bricklin's Demo Program Prototype quickly. User feedback without programming. All 250 ASCII characters plus attributes. Subsetting, macros. PC $ 75

BetterBASIC - all RAM, modules, structure. **Full BASICA**	PC	$169
8087 Math Support	PC	$ 89
Run-time module	PC	$235

Free Literature Compare Products

Evaluate products. *Compare* competitors. Learn about new alternatives. One *free call* brings information on just about any programming need. Ask for any ***"Packet"*** or Addon Packet □ AI □ ADA, Modula □ BASIC □"C" □ COBOL □ Editors □ FORTH □ FORTRAN □ PASCAL □ UNIX/PC or □ Debuggers, Linkers.

Our Services:

- **Programmer's Referral List**
- **Compare Products**
- **Help find a Publisher**
- ***Evaluation Literature FREE***
- **BBS - 7 PM to 7 AM 617-826-4086**
- **Dealers Inquire**
- **Newsletter**
- **Rush Order**
- **Over 700 products**
- **National Accounts Center**

BASIC

ACTIVE TRACE Debugger - BASICA, MBASIC, well liked	MS	$ 79
APC MegaBASIC - powerful	PC	$339
BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum. crossref, compress.	PC	$105
Basic Windows by Syscom	PC	$ 95
CADSAM FILE SYSTEM - full ISAM in MBASIC source.	MS	$ 75
Prof. Basic - Interactive, debug	PC	$ 79
8087 Math Support	PC	$ 47
QuickBASIC by Microsoft - Compiles full IBM BASICA, 640K	PC	$ 79
TRUE Basic - ANSI	PC	$119
Run-time Module	PC	$459

Cobol

Macintosh COBOL - full	MAC	$459
MBP - Lev. II, native	MS	$885
MicroFocus Professional - full	PC	Call
Microsoft Version II - upgraded. Full Lev. II, native, screens.	MS	$495
Realia - very fast	MS	$929
Ryan McFarland - portable	MS	$699

Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure	PC	Call
C Screen with source	80/86	$ 75
EMACS by UniPress - powerful, multifile, windows	Source:	$949 $299
Epsilon - like EMACS, full C-like language for macros.	PC	$169
FirsTime by Spruce - Improve productivity. Syntax directed for Turbo ($69), Pascal ($229), or C ($239)		
Kedit - like XEDIT	PC	$115
Lattice Screen Editor - multiwindow, multitasking	Amiga $100	MS $125
PMATE - power, multitask	80/86	$159
VEDIT - well liked, macros, buffers	CPM-80-86	MS $119
XTC - multitasking	PC	$ 85

Atari ST & Amiga

We carry full lines of Manx, Lattice, Metacompco & Prospero.

Cambridge LISP	Amiga	$200
Lattice C	Atari ST	$139
Lattice Text Utilities	Amiga	$ 75
LINT by Gimpel	Amiga	$ 89
Megamax - tight, full	Atari ST	$179

RECENT DISCOVERY

dBASE Tools for C - incorporate C functions as extensions to dBASE III Plus. Also functions for business graphics, arrays, math, statistics. MS C, Lattice Aztec. PC Graphics $ 79 Tools $ 79

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
AZTEC C85 - Personal	Apple II	$199
C86 by CI - 8087, reliable	MS	$299
Consulair Mac C w/toolkit	MAC	$299
Lattice C - from Lifeboat	MS	$289
Lattice C - from Lattice	MS	$339
Mark Williams - w/debugger	MS	$399
Microsoft C 3.0 - new	MS	$259
Q/C 88 by Code Works - Compiler source, decent code, native	MS	$125
Wizard C - Lattice C compatible, full sys. III, lint, fast.	MS	$389

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$249
INSTANT C - Source debug, Edit to Run-3 seconds	MS	$399
Interactive C by IMPACC Assoc. Interpreter, editor, source, debug.	PC	$225
Introducing C - self paced tutorial	PC	$109
Professional Run/C - Run/C plus create add-in libraries, more	MS	$199
Run/C - improved	MS	$109

C Libraries-General

Application Programmer's Toolkit	MS	$375
Blaise C Tools 1 ($109), C Tools 2		$ 89
C Food by Lattice-ask for source	MS	$109
C Utilities by Essential - Comprehensive screen graphics, strings, source.	PC	$139
C Worthy Library - Complete, machine independent, source	MS	$295
Entelekon C Function Library	PC	$119
Entelekon Superfonts for C	PC	$ 45
Greenleaf Functions - portable, ASM		$139
Polytron - for Lattice, ASM source		$ 85
Software Horizons - Pack I	PC	$129

C Libraries-Communications

Asynch by Blaise	PC	$149
Greenleaf - full, fast	PC	$139
Software Horizons - pack 3	PC	$119

C Libraries-Files

FILES: C Index by Trio - full B + Tree, vary length field, multi compiler		
/File is object only	MS	$ 89
/Plus is full source	MS	$349
C to dBase - with source	MS	$139
CBTREE - multiuser record locking, sequential, source, no royalties	MS	$ 99
dbc Isam by Lattice	MS	$199
dbVISTA - full indexing, plus optional record types, pointers, Network.		
Object only - MS C, LAT,	C86	$159
Source - Single user	MS	$429
Source - **Multiuser**	MS	$849

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

Circle **no. 141** on reader service card.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

ORDER TODAY SPECIAL PRICE

C-TREE by Faircom - portable, stable, thorough, balanced Btree file manager with source. MSDOS List $395
ORDER BEFORE 6/30/86 and mention this ad for a special price of $289.
ORDER AFTER 6/30/86 for $349.

C Support-Systems

Basic-C Library by C Source	PC	$139
CPRINT - by ENSCO	MS	$ 45
C Sharp - well supported. Source, **realtime,** tasks	MS	$600
C ToolSet - DIFF, xref, source	MS	$135
Lattice Text Utilities	MS	$105
The HAMMER by OES Systems	PC	$179
PC LINT - Checker	MS	$125
SECURITY LIB - add encrypt to C86 programs. Source $250	MS C, PC	$125

C-Screens, Windows, Graphics

Curses by Lattice	PC	$109
C Power Windows by Entelekon	PC	$119
C Windows by Syscom	PC	$ 89
ESSENTIAL GRAPHICS - fast, fonts, no royalties	PC	$219
GraphiC - source in C	PC	$219
Topview Toolbasket by Lattice	PC	$209
View Manager for C by Blaise	PC	$219
Windows for C - fast	PC	$149
Windows for Data - validation	PC	$259

Debuggers

Advanced Trace-86 by Morgan Modify ASM code on fly.	PC	$149
CODESMITH - visual, modify and rewrite Assembler	PC	$109
C SPRITE - data structures	PC	$139
Periscope I - own 16K	PC	$269
Periscope II - symbolic, "Reset Box," 2 Screen	PC	$129
Pfix-86 Plus Symbolic Debugger by Phoenix - windows	PC	$289
Software Source by Atron - Lattice, MS C, Pascal, Windows single step, 2 screen, log file.	MS	$115
w/Breakswitch		$199

FEATURES

dBrief, the dBASE Assistant - optional syntax directed editing, screen gen, graphics, speed coding. dBASE II, III. Clipper. PC $ 95

Microsoft Cobol Tools - symbolic, windowing debugger w/source support. Plus cross reference. Menu Handler, mouse support. PC $310

Fortran & Supporting

ACS Time Series	MS	$469
Forlib + by Alpha - graphics and file routines, comm.	MS	$ 59
MACFortran by Microsoft - full '77. Includes ASM output	MAC	$229
MS Fortran	MS	**$219**
No Limit - Fortran Scientific	PC	$129
PolyFortran - xref, pp, screen	MS	$149
Prospero - '66, reentrant	MS	$349
RM Fortran - enhanced "IBM Professional Fortran"	MS	$399
Scientific Subroutines - Matrix	MS	$149
Statistician by Alpha	MS	$269
Strings and Things - register, shell	PC	$ 59

MultiLanguage Support

BTRIEVE ISAM	MS	$199
BTRIEVE/N-multiuser	MS	$469
CODESIFTER - Execution PROFILER. Spot bottlenecks.	MS	$109
HALO Graphics- Multiple video boards, printer, rich. Animation, engineering, business. Any MS language, Lattice, C86	PC	$249
Panel - Create screen with editor, generates code. Full data validation, no royalties.	MS	$239
PLINK 86 - a program-independent overlay linker to 32 levels.	MS	$279
PLINK-86 PLUS - incremental	MS	$369
Pfinish Performance Analyzer	PC	$279
Profiler by DWB Associates	MS	$ 99
PolyLibrarian by Polytron	MS	$ 85
PVCS Version Control	MS	$359
Screen Sculptor - slick, thorough, fast, BASIC, PASCAL.	PC	$ 99
ZAP Communications - VT 100, TEK 4010 emulation, full xfer.	PC	$ 85

Turbo PASCAL & Supporting

BORLAND: Turbo 3.0	$ 49
3.0 with 8087 or BCD	$ 79
3.0 with 8087 and BCD	$ 85
Turbo Graphix - graphs, windows	$ 39
Turbo Toolbox or Editor	$ 55
Turbo Tutor	$ 29
TURBO . . . Asynch by Blaise, full	$ 85
Power Tools by Blaise - library	$ 85
Power Utilities - profiler, pp	$ 85
Professional - interrupts, macros.	$ 50
OTHERS: Screen Sculptor ($99), Pascal Pac ($100), Tidy ($45).	

RECENT DISCOVERY

Synergy - Create user interfaces. TopView-compatible multitasking operating environment offers windows, icons, pull-down menus and fonts in 12K RAM. MS $375

Other Languages

APL + /PC	PC	$469
CLIPPER-dBASE Compiler	MS	$449
ED/ASM-86 by Oliver	PC	$ 85
HS/FORTH - '79 & '83 Standards, full RAM, ASM, BIOS, interrupts. Graph, multi-task, optimizer	MS	$250
MacASM - fast	MAC	$ 99
MasterForth	MAC or PC	$125
Microsoft MASM - faster	MS	$109
Microsoft PASCAL	MS	$199
MODULA: M2SDS - popular	PC	$ 69
MICROTEC PASCAL - extensions like packages, "Iterators", 5 memory models. 65 bit 8087 strings.	MS	$665
Pasm - by Phoenix	MS	$219
Prospero Pascal - full ISO+	MS	$349
RPG II by Lattice	PC	$675
SNOBOL4 + - great for strings	MS	$ 85
Turbo Edit/ASM - by Speedware	PC	$ 85

Xenix-86 & Supporting

Basic - by Microsoft	$279
Cobol - by Microsoft	$795
Fortran - by Microsoft	$399
PANEL Screen LIB - multi-language	$539
Xenix Complete Development System	$985

Other Products

dBASE to C Translator: dBx - no royalties, addon ISAM, **Pioneer it.** Source $1000	MS	$ 350
HTest/H Format - XT Fix	PC	$ 119
Microsoft Windows	PC	$ 75
Opt Tech Sort- sort, merge	MS	$ 135
Polymake by Polytron	MS	$ 79
PS MAKE - Directly execute or Gen a batch file, interactive.	MS	$ 129
Qwik Net - critical path, 125 tasks/ resources, thorough; usable	PC	$ 316
SECRET DISK by Lattice	PC	$ 49
SET: SCIL	MS	$ 319
SoftEst - Softwear Estimating and reporting. Pioneer it.	MS	$ 350
Texsys - control source	MS	$ 89
Visible Computer: 8088 - Simulates demos or any .exe. com, Debugger. 350 pg. tutorial	PC	$ 59

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. All formats available. UPS surface shipping add $3/item.

Call for a catalog, literature, advice and service you can trust

NEW HOURS
8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™
128-D Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531 486

"I have been pleased with your catalog selection, the knowledge of your telephone answering staff, and the promptness of your service. I have every wish to become a continuing customer of the Programmer's Shop.

Carl C. Rollo

Circle **no. 133** on reader service card.

Listing One (Text begins on page 38.)

```
/**
 * Copyright (c) 1985 by Steve Wilhite, Worthington, Ohio
 *
 * Permission is granted to use or distribute this software without any
 * restrictions as long as this entire copyright notice is included intact.
 * You may include it in any software product that you sell for profit.
 *
 * This software is distributed as is, and is not guaranteed to work on any
 * given hardware/software configuration.  Furthermore, no liability is
 * granted with this software.
 *
 * ABSTRACT:
 *
 *       The function, Transfer_File, implements error-free file transfer using
 *       CompuServe's "B" protocol.
 *
 *       It has been assumed that the start-of-packet sequence, DLE "B", has
 *       been detected and the next byte not received yet is the packet
 *       sequence number (an ASCII digit).
 *
 * ENVIRONMENT: Lattice "C", machine independent.
 *
 * AUTHOR: Steve Wilhite, CREATION DATE: 21-Jul-85
 *
 * REVISION HISTORY:
 *
 *       Steve Wilhite, 17-Jan-86
 *       - included a virtual file interface.
 **/

/*** Feature Test ***/

/* Strip_CR and Strip_LF are mutual exclusive!! */

#define Strip_CR    0         /* If true, strip CR's before writing to disk.
                                 Add CR before sending */
#define Strip_LF    0         /* If true, strip LD's before writing to disk.
                                 Add LF before sending */

/* External Functions */

extern Delay();                         /* Sleep for "n" milliseconds */
extern Put_Char();              /* Write a character to the display */
extern Start_Timer();                   /* Enable the timer for the specified number
                                   seconds */
extern int Timer_Expired();  /* Returns "true" if the timer has expired,
                                   "false" otherwise */
extern int Wants_To_Abort(); /* Returns "true" if the user wants to abort
                                   the file transfer, "false" otherwise */
extern int Read_Modem();     /* Read a character from the comm port.
                                   Returns -1 if no character available */
extern int Write_Modem();    /* Send a character to the comm port. Returns
                                   "true" is successful, "false" otherwise */

/* File I/O Interface */

extern int Create_File(), Open_File(), Close_File();
extern int Read_File(), Write_File();


#define NUL         0x00
#define ETX         0x03
#define ENQ         0x05
#define DLE         0x10
#define XON         0x11
#define XOFF        0x13
#define NAK         0x15

#define True                1
#define False               0
#define Success             -1
#define Failure             0
#define Packet_Size 512
#define Max_Errors  10
#define Max_Time    10
#define Max_Xoff_Time       10
#define WACK                ';'             /* wait acknowledge */

/* Sender actions */

#define S_Send_Packet       0
#define S_Get_DLE   1
#define S_Get_Num   2
#define S_Get_Seq   3
#define S_Get_Data  4
#define S_Get_Checksum      5
#define S_Timed_Out 6
#define S_Send_NAK  7
#define S_Send_ACK  8

/* Receiver actions */

#define R_Get_DLE   0
#define R_Get_B             1
#define R_Get_Seq   2
#define R_Get_Data  3
#define R_Get_Checksum      4
```

(continued on page 92)

Now You Know Why BRIEF™ is BEST

"BRIEF has improved my productivity tenfold. It paid for itself in 2 weeks!"

David Norwood, Microsystems Manager

The Program Editor with the BEST Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)...is quite simply the best code editor I have seen."

REGULAR EXPRESSION SEARCH

Regular expression searching is one of BRIEF's most powerful features. A regular expression is a series of "wildcards" that match pieces of your text. BRIEF supports a full set of regular expression characters similar to those found in UNIX including: beginning an end of line, groups, and the "closure" and "or" operators.

As Steve McMahon explained in Byte, "Not only does BRIEF make use of this marvelously general regular expression notation in its search facility, but its pattern recognition extends to its replacement (or translation) facility." "The usefulness of this facility for programmers who deal constantly with the regular expressions of formal languages is obvious . . ."

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size –(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

Program Editing YOUR Way

A typical program editor requires you to adjust your style of programming to its particular requirements – NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key – even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion – BRIEF IS BEST!

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

Solution Systems™

MONEY-BACK GUARANTEE
Try BRIEF ($195) for 30 days – If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

SOLUTION SYSTEMS, 335-D WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

BRIEF is a trademark of UnderWare

Circle **no. 147** on reader service card.

Listing One (Listing continued, text begins on page 38.)

```
#define R_Send_NAK 5
#define R_Send_ACK 6

static int
    Ch,
    Checksum,
    Seq_Num,
    R_Size,                                    /* Size of receiver buffer */
    XOFF_Flag,
    Seen_ETX,
    Seen_ENQ;

static char
    S_Buffer[Packet_Size],          /* Sender buffer */
    R_Buffer[Packet_Size];          /* Receiver buffer */


static Put_Msg(Text)
    char *Text;
    {
    while (*Text != 0)
         Put_Char(*Text++);

    Put_Char('\015');
    Put_Char('\012');
    }


static Send_Byte(Ch)
    int Ch;
    {
    int TCh;

    /* Listen for XOFF from the network */

    Start_Timer(Max_Xoff_Time);
    do
         {
         while ((TCh = Read_Modem()) >= 0)
             if (TCh == XON)
                  XOFF_Flag = False;
             else if (TCh == XOFF)
                  {
                  XOFF_Flag = True;
                  Start_Timer(Max_Xoff_Time);
                  }
         }
    while (XOFF_Flag && !Timer_Expired());

    while (!Write_Modem(Ch));
    }


static Send_Masked_Byte(Ch)
    int Ch;
    {
    /* Mask any protocol or flow characters */

    if (Ch == NUL || Ch == ETX || Ch == ENQ || Ch == DLE || Ch == NAK || Ch == XON || Ch == XOFF)
         {
         Send_Byte(DLE);
         Send_Byte(Ch + '@');
         }
    else
         Send_Byte(Ch);
    }


static Send_ACK()
    {
    Send_Byte(DLE);
    Send_Byte(Seq_Num + '0');
    }


 static Read_Byte()
     {
     if ((Ch = Read_Modem()) < 0)
          {
          Start_Timer(Max_Time);

          do
              {
              if (Timer_Expired())
                   return Failure;
              }
          while ((Ch = Read_Modem()) < 0);
          }

     return Success;
     }


 static Read_Masked_Byte()
     {
```

(continued on page 94)

MS-DOS, UNIX, APPLE MAC, CP/M, NETWORKS and MORE. ONE c-tree ISAM DOES THEM ALL!

The creator of Access Manager™ brings you the most powerful C source code, B+ Tree file handler: c-tree™

- multi-key ISAM and low-level B+ Tree routines
- complete C source code written to K&R standards
- single-user, network and multi-tasking capabilities
- fixed and variable record length data files
- virtually opened files accommodate limited file descriptors
- no royalties on application programs

$395 COMPLETE

Specify diskette format:
- 5¼" MS-DOS
- 8" CP/M
- 3½" Mac
- 8" RT-11

For VISA, MC and COD orders call (314) 445-6833
FairCom
2606 Johnson Drive
Columbia, MO 65203

© 1985 FairCom

The following are trademarks: c-tree and the circular disk logo—FairCom; MS—Microsoft Inc.; CP/M and Access Manager—Digital Research Inc.; Unix—AT&T; Apple—Apple Computer Co.

Circle **no. 93** on reader service card.

PLOTDEV ADDS GRAPHICS TO PC-DOS $39

MicroPlot, innovator of creative PC software programs, is excited to announce PlotDev - the alternative to virtual device drivers and graphics tool kits for PC-DOS graphics programs.

- **Installable PC-DOS device driver adds graphics command capabilities to any program language.**
- **Allows full screen PC-DOS command editing by unsticking the cursor.**
- **Supports most popular graphics boards and provides a built-in graphics screen dump for printers.**
- **Non-interfering with memory resident or application programs.**
- **Provides user with all the intelligent alpha terminal commands of a DEC VT-100.**
- **Provides user with the graphics commands of the Tektronix 4010/4014 and 4027 graphics terminals.**
- **And much more ...**

For a detailed PlotDev brochure or for ordering information call toll free 1-800-338-0333, in Ohio call 1-800-242-0333. Use touch-tone to enter I.D. code 766-8501, or wait for operator assistance. Visa or MasterCard welcomed.

MicroPlot™

659-H Park Meadow Road
Westerville, Ohio 43081 (614) 882-4786

Site license and educational discounts available.

Circle **no. 94** on reader service card.

The Answer to your Debugging Problems

ICD286

At last! An 80286 emulator which is affordable, compact, and easy to use.

FAST—Full speed, real-time emulation up to 10 Mhz.

AFFORDABLE—From $2400 to $5400, depending on options.

EASY TO USE—On-line HELP with a screen oriented display.

KEY FEATURES:

- Hardware and software breakpoints
- 2048 bus cycles of real-time trace
- 64K of emulation memory
- Symbol and line number support
- Source-level debugging
- Real and virtual (protected) mode support
- On-line symbolic assembly and disassembly
- Macros with parameters
- Installs in an IBM-PC/XT/AT* or compatible

IDEAL for development, debugging, testing, and field service.

For further information, please contact:

Answer Software
Corporation
20863 Stevens Creek Boulevard
Cupertino, CA 95014
(408) 253-7515

*IBM-PC/XT/AT are registered trademarks of International Business Machines Corporation.

Circle **no. 171** on reader service card.

AT LAST: Professional Typesetting Capability For PC Users

With PC TEX™ — the best-selling full implementation of Professor Don Knuth's revolutionary typesetting program TEX.

FINEST Typeset Quality Printing From:
dot matrix laser phototypesetter

$$\sum_{i=1}^{\infty} \frac{1}{i} \qquad \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \qquad \int_{-\infty}^{\infty} e^{-x^2}\,dx$$

WIDEST Range Of Output Device Drivers:

- Epson FX, LQ
- Toshiba
- Corona LP-300*
- Screen preview, with EGA or Hercules card
- HP LaserJet*
- Apple LaserWriter
- APS-5 phototypesetter

MOST COMPLETE Product Offering:

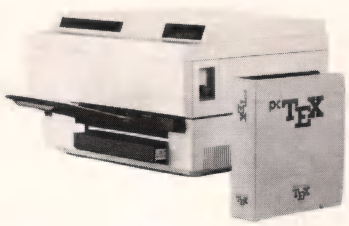
PC TEX (not copy protected) includes the following:

- Our specially written *PC TEX Manual*, which enables you to start using TEX right away.
- Custom "macro packages" that provide formats for letters, manuals, technical documents, etc.
- The LATEX document preparation system, a full-featured macro package for preparing articles, books, reports, etc., and LATEX User's Manual.
- AMS-TEX, developed by the *Amer. Math. Society* for professional mathematical typesetting.

Site licenses, volume discounts, and interfaces to PC Paintbrush, PC Palette, FancyFont and Fontrix are also available.

PRICED FROM ONLY $249.00!

(Printer drivers and interfaces additional.)



Laser printer, fonts & software from $2995.00

For IBM PC/XT, AT or compatible, DOS 2.0 or higher, and 512K RAM. Hard disk required for printer drivers and fonts. *HP LaserJet and Corona require additional interface boards.

For more information call or write:

Personal TEX, Inc.

20 Sunnyside, Suite H, Mill Valley, CA 94941 (415) 388-8853

This ad, with space for the photograph, produced by PC TEX. Typeset on the Epson FX80, the Corona LP-300 laser printer, and the Autologic APS-5 phototypesetter.

TEX is a trademark of the American Mathematical Society. Manufacturers' product names are trademarks of individual manufacturers.

Circle **no. 76** on reader service card.

B PROTOCOL

Listing One

(Listing continued, text begins on page 38.)

```
    Seen_ETX = False;
    Seen_ENQ = False;

    if (Read_Byte() == Failure)
          return Failure;

    if (Ch == DLE)
          {
          if (Read_Byte() == Failure)
              return Failure;

          Ch &= 0x1F;
          }
    else if (Ch == ETX)
          Seen_ETX = True;
    else if (Ch == ENQ)
          Seen_ENQ = True;

    return Success;
    }


static Do_Checksum(Ch)
    int Ch;
    {
    Checksum <<= 1;

    if (Checksum > 255)
          Checksum = (Checksum & 0xFF) + 1;

    Checksum += Ch & 0xFF;

    if (Checksum > 255)
          Checksum = (Checksum & 0xFF) + 1;
    }


static int Read_Packet(Action)
/**
 * Function:
 *       Receive a packet from the host.
 *
 * Inputs:
 *       Action -- the starting action
 *
 * Outputs:
 *       R_Buffer -- contains the packet just received
 *       R_Size -- length of the packet
 *
 * Returns:
 *       success/failure
 **/
    int Action;
    {
    int
          Errors,
          Next_Seq;

    Errors = 0;

    while (Errors < Max_Errors)
          switch (Action)
              {
              case R_Get_DLE:
                    if (Read_Byte() == Failure)
                        Action = R_Send_NAK;
                    else if (Ch == DLE)
                        Action = R_Get_B;
                    else if (Ch == ENQ)
                        Action = R_Send_ACK;

                    break;

              case R_Get_B:
                    if (Read_Byte() == Failure)
                        Action = R_Send_NAK;
                    else if (Ch == 'B')
                        Action = R_Get_Seq;
                    else
                        Action = R_Get_DLE;

                    break;

              case R_Get_Seq:
                    if (Read_Byte() == Failure)
                        Action = R_Send_NAK;
                    else
                        {
                        Checksum = 0;
                        Next_Seq = Ch - '0';
                        Do_Checksum(Ch);
                        R_Size = 0;
                        Action = R_Get_Data;
                        }
```

```
                break;

            case R_Get_Data:
                if (Read_Masked_Byte() == Failure)
                    Action = R_Send_NAK;
                else if (Seen_ETX)
                    Action = R_Get_Checksum;
                else if (Seen_ENQ)
                    Action = R_Send_ACK;
                else if (R_Size == Packet_Size)
                    Action = R_Send_NAK;
                else
                    {
                    R_Buffer[R_Size++] = Ch;
                    Do_Checksum(Ch);
                    }

                break;

            case R_Get_Checksum:
                Do_Checksum(ETX);

                if (Read_Masked_Byte() == Failure)
                    Action = R_Send_NAK;
                else if (Checksum != Ch)
                    Action = R_Send_NAK;
                else if (Next_Seq == Seq_Num)
                    Action = R_Send_ACK;    /* Ignore duplicate packet */
                else if (Next_Seq != (Seq_Num + 1) % 10)
                    Action = R_Send_NAK;
                else
                    {
                    Seq_Num = Next_Seq;

                    return Success;
                    }

                break;

            case R_Send_NAK:
                Put_Char('-');
                Errors++;
                Send_Byte(NAK);
                Action = R_Get_DLE;
                break;

            case R_Send_ACK:
                Send_ACK();
                Action = R_Get_DLE;
                break;
            }

    return Failure;
    }

static int Send_Packet(Size)
/**
 * Function:
 *      Send the specified packet to the host.
 *
 * Inputs:
 *      Size -- length of the packet
 *      S_Buffer -- the packet to send
 *
 * Outputs:
 *
 * Returns:
 *      success/failure
 **/
    int Size;
    {
    int
        Action,
        Next_Seq,
        RCV_Num,
        I,
        Errors;

    Next_Seq = (Seq_Num + 1) % 10;
    Errors = 0;
    Action = S_Send_Packet;

    while (Errors < Max_Errors)
        switch (Action)
            {
            case S_Send_Packet:
                Checksum = 0;
                Send_Byte(DLE);
                Send_Byte('B');
                Send_Byte(Next_Seq + '0');
                Do_Checksum(Next_Seq + '0');

                for (I = 0; I < Size; I++)
                    {
                    Send_Masked_Byte(S_Buffer[I]);
                    Do_Checksum(S_Buffer[I]);
                    }

                Send_Byte(ETX);
                Do_Checksum(ETX);
                Send_Masked_Byte(Checksum);
```

(continued on next page)

Put More UNIX™ in Your C.

Unitools $99
MAKE, DIFF and GREP
These versatile UNIX-style utilities put power at your fingertips. MAKE, a program administrative tool, is like having an assistant programmer at your side. DIFF compares files and shows you the differences between them. GREP can search one or many files looking for one pattern or a host of patterns.

"Z" $99
A Powerful "vi"-type Editor.
Similar to the Berkeley "vi" editor, "Z's" commands are flexible, terse, and powerful; macro functions give you unlimited range. Features include "undo," sophisticated search and replace functions, automatic indentation, C-tags, and much, much more.

PC-LINT $99
Error Checking Utility
A LINT-like utility that analyzes programs and uncovers bugs, quirks and inconsistencies. Detects subtle errors. Supports large and small memory models, has clear error messages and executes quickly. Has lots of options and features that you wouldn't expect at this low price.

SunScreen $99
Low-priced Screen Utility
This versatile graphics package easily creates and modifies formatted screens, validates fields, supports function keys, color and monochrome cards. With library source SunScreen is $199.

Compatible with all leading MS/PC-DOS C compilers.

SPECIAL OFFER:
Unitools, "Z," PC-LINT and SunScreen All for only $349

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)

UNIX is a registered TM of Bell Laboratories. MANX AZTEC TM Manx Software Systems, Inc. PC LINT TM GIMPLE software. SunScreen TM SunTec. MS-DOS TM Microsoft

Circle **no. 223** on reader service card.

Listing One (Listing continued, text begins on page 38.)

```
        Action = S_Get_DLE;
        break;

    case S_Get_DLE:
        if (Read_Byte() == Failure)
            Action = S_Timed_Out;
        else if (Ch == DLE)
            Action = S_Get_Num;
        else if (Ch == ENQ)
            Action = S_Send_ACK;
        else if (Ch == NAK)
            {
            Errors++;
            Action = S_Send_Packet;
            }

        break;

    case S_Get_Num:
        if (Read_Byte() == Failure)
            Action = S_Timed_Out;
        else if (Ch >= '0' && Ch <= '9')
            {
            if (Ch == Seq_Num + '0')
                Action = S_Get_DLE; /* Ignore duplicate ACK */
            else if (Ch == Next_Seq + '0')
                {
                /* Correct sequence number */

                Seq_Num = Next_Seq;
                return Success;
                }
            else if (Errors == 0)
                Action = S_Send_Packet;
            else
                Action = S_Get_DLE;
            }
        else if (Ch == WACK)
            {
            Delay(5000);    /* Sleep for 5 seconds */
            Action = S_Get_DLE;
            }
        else if (Ch == 'B')
            Action = S_Get_Seq;
        else

            Action = S_Get_DLE;

        break;

    case S_Get_Seq:
        /**
         * Start of a "B" protocol packet. The only packet that makes
         * any sense here is a failure packet.
         **/

        if (Read_Byte() == Failure)
            Action = S_Send_NAK;
        else
            {
            Checksum = 0;
            RCV_Num = Ch - '0';
            Do_Checksum(Ch);
            I = 0;
            Action = S_Get_Data;
            }

        break;

    case S_Get_Data:
        if (Read_Masked_Byte() == Failure)
            Action = S_Send_NAK;
        else if (Seen_ETX)
            Action = S_Get_Checksum;
        else if (Seen_ENQ)
            Action = S_Send_ACK;
        else if (I == Packet_Size)
            Action = S_Send_NAK;
        else
            {
            R_Buffer[I++] = Ch;
            Do_Checksum(Ch);
            }

        break;

    case S_Get_Checksum:
        Do_Checksum(ETX);

        if (Read_Masked_Byte() == Failure)
            Action = S_Send_NAK;
        else if (Checksum != Ch)
            Action = S_Send_NAK;
        else if (RCV_Num != (Next_Seq + 1) % 10)
            Action = S_Send_NAK;
        else
            {
```

```
                        /**
                         * Assume the packet is failure packet. It makes no
                         * difference since any other type of packet would be
                         * invalid anyway. Return failure to caller.
                         **/

                        Errors = Max_Errors;
                        }

                    break;

                case S_Timed_Out:
                    Errors++;
                    Action = S_Get_DLE;
                    break;

                case S_Send_NAK:
                    Put_Char('-');
                    Errors++;
                    Send_Byte(NAK);
                    Action = S_Get_DLE;
                    break;

                case S_Send_ACK:
                    Send_ACK();
                    Action = S_Get_DLE;
                    break;
                }

        return Failure;
        }

static Send_Failure(Code)
/**
 * Function:
 *      Send a failure packet to the host.
 *
 * Inputs:
 *      Code -- failure code
 *
 * Outputs:
 *
 * Returns:
 **/
    char Code;
    {
    S_Buffer[0] = 'F';
    S_Buffer[1] = Code;
    Send_Packet(2);
    }

static int Receive_File(Name)
/**
 * Function:
 *      Download the specified file from the host.
 *
 * Inputs:
 *      Name -- ptr to the file name string
 *
 * Outputs:
 *
 * Returns:
 *      success/failure
 **/
    char *Name;
    {
    int Data_File;                  /* file descriptor */

    if ((Data_File = Create_File(Name, 0)) == -1)
        {
        Put_Msg("Cannot create file");
        Send_Failure('E');
        return Failure;
        }

    Send_ACK();

    for (;;)
        if (Read_Packet(R_Get_DLE) == Success)
            switch (R_Buffer[0])
                {
                case 'N':           /* Data packet */

                    if (Write_File(Data_File, &R_Buffer[1], R_Size - 1) != R_Size - 1)
                        {
                        /* Disk write error */

                        Put_Msg("Disk write error");
                        Send_Failure('E');
                        Close_File(Data_File);
                        return Failure;
                        }

                    if (Wants_To_Abort())
                        {
                        /* The user wants to kill the transfer */

                        Send_Failure('A');
                        Close_File(Data_File);
                        return Failure;
                        }
```

(continued on next page)

C Users' Group

Over 85 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group
Post Office Box 97
McPherson, KS 67460
(316) 241-1065

Circle **no. 181** on reader service card.

IBM PC
CROSS
ASSEMBLERS

Assemblers now available include:

Chip	Chip	Chip
1802/1805	HD64180	8051
9900/9995	NSC800	6804
6500/01/02	F8, 3870	6805
6800/01/02	Z8	6809
6800/08/10	Z80	6811
8048/49/50/42	Z8000	8085
65C02/C102/C112	6803/08	6301

RELATIONAL MEMORY SYSTEMS, INC.
P. O. Box 6719
San Jose, California 95150

Tel: (408) 265-5411
CPM80, MPM, ISIS Versions Available

relms®

Circle **no. 120** on reader service card.

Gary Kildall (CPM)

"Somehow we have to break loose from the ways we think about microcomputers if we want to stimulate advances in computers."

Programmers at Work

Interviews with 19 of Today's Top Programmers
$14.95 (soft) **$19.95** (hard)
Available wherever fine books are sold.

MICROSOFT PRESS

Circle **no. 125** on reader service card.

Listing One (Listing continued, text begins on page 38.)

```
                Send_ACK();
                Put_Char('+');
                break;

            case 'T':           /* Transfer packet */

                if (R_Buffer[1] == 'C') /* Close file */
                    {
                    Send_ACK();
                    Close_File(Data_File);
                    return Success;
                    }
                else
                    {
                    /**
                     * Unexpected "T" packet. Something is rotten on the
                     * other end. Send a failure packet to kill the
                     * transfer cleanly.
                     **/

                    Put_Msg("Unexpected packet type");
                    Send_Failure('E');
                    Close_File(Data_File);
                    return Failure;
                    }

            case 'F':           /* Failure packet */
                Send_ACK();
                Close_File(Data_File);
                return Failure;
            }
    else
        {
        Close_File(Data_File);
        return Failure;
        }
}

static int Send_File(Name)
/**
 * Function:
 *      Send the specified file to the host.
 *
 * Inputs:
 *      Name -- ptr to the file name string
 *
 * Outputs:
 *
 * Returns:
 *      success/failure
 **/
    char *Name;
    {
    int
        Data_File,                          /* file descriptor */
        N;

    if ((Data_File = Open_File(Name, 0)) == -1)
        {
        Put_Msg("Cannot access that file");
        Send_Failure('E');
        return Failure;
        }

    do
        {
        S_Buffer[0] = 'N';
        N = Read_File(Data_File, &S_Buffer[1], Packet_Size - 1);

        if (N > 0)
            {
            if (Send_Packet(N + 1) == Failure)
                {
                Close_File(Data_File);
                return Failure;
                }

            if (Wants_To_Abort())
                {
                Send_Failure('A');
                Close_File(Data_File);
                return Failure;
                }

            Put_Char('+');
            }
        }
    while (N > 0);

    if (N == 0)                             /* end of file */
        {
        Close_File(Data_File);
        S_Buffer[0] = 'T';
        S_Buffer[1] = 'C';
        return Send_Packet(2);
        }
```

```
    else
        {
        Put_Msg("Disk read error");
        Send_Failure('E');
        return Failure;
        }
    }

int Transfer_File()
/**
 * Function:
 *      Transfer a file from/to the micro to/from the host.
 *
 * Inputs:
 *
 * Outputs:
 *
 * Returns:
 *      success/failure
 **/
    {
    int I, N;
    char Name[64];                      /* holds the file name */

    XOFF_Flag = False;
    Seq_Num = 0;

    if (Read_Packet(R_Get_Seq) == Success)
        {
        if (R_Buffer[0] == 'T')                 /* transfer packet */
            {
            /* Check the direction */

            if (R_Buffer[1] != 'D' && R_Buffer[1] != 'U')
                {
                Send_Failure('N'); /* not implemented */
                return Failure;
                }

            /* Check the file type */

            if (R_Buffer[2] != 'A' && R_Buffer[2] != 'B')
                {
                Send_Failure('N');
                return Failure;
                }

            /* Collect the file name */

            N = R_Size - 3 > 63 ? 63 : R_Size - 3;

            for (I = 0; I < N; I++)
                Name[I] = R_Buffer[I + 3];

            Name[I] = 0;

            /* Do the transfer */

            if (R_Buffer[1] == 'U')
                return Send_File(Name);
            else
                return Receive_File(Name);
            }
        else
            {
            Send_Failure('E');                  /* wrong type of packet */
            return Failure;
            }
        }
    else
        return Failure;
    }
```

End Listing One

Listing Two

```
        title   Keyboard Driver
        include \lc\dos.mac
        pseg

        public  Read_Keyboard

Read_Keyboard proc
;++
; Function:
;       Read a "raw" character from the keyboard.
;
; Inputs: none
;
; Outputs: none
;
; Returns:
;       -1 if no character is available; otherwise a 16-bit code.
;       If the high byte is zero, then the low byte is an ASCII character,
;       else the low byte is an "extended" character (scan code).
;--
        mov     AH,1
```

(continued on next page)

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN

OUTSIDE OKLAHOMA: NO SALES TAX

640 Kbyte MOTHERBOARD KITS: Zenith 150: $87.46
IBM PC/XT, Compaq Portable & Plus/HP Vectra

DYNAMIC RAM				
256K	64Kx4	150 ns	$4.85	
256K	256Kx1	100 ns	6.20	
256K	256Kx1	120 ns	3.90	
256K	256Kx1	150 ns	3.47	8 Mhz $ 17.50
128K	128Kx1	150 ns	4.92	8 Mhz $375.00
64K	64Kx1	150 ns	1.60	
EPROM				
27512	64Kx8	250 ns	$30.00	V20/V30
27C256	32Kx8	250 ns	8.15	80287
27256	32Kx8	250 ns	5.45	
27128	16Kx8	250 ns	3.90	
27C64	8Kx8	200 ns	5.30	8087-2 $185.00
2764	8Kx8	250 ns	3.80	8087-3 $135.00
2732	4Kx8	450 ns	3.85	
STATIC RAM				
6264LP-15	8Kx8	150 ns	$3.45	
6116LP-3	2Kx8	150 ns	2.10	

OPEN 6½ DAYS: WE CAN SHIP VIA FED-EX ON SAT.

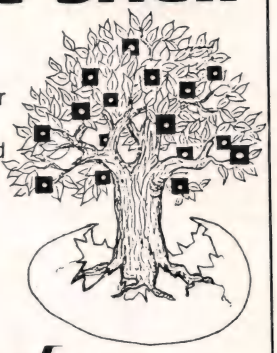
NO EXTRA COST FOR FED-EX SAT DELIVERY ON ORDERS RECEIVED BY
Th: Std Air $6/4 lbs
Fr: P-One $13/2 lbs

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts µP∞
MICROPROCESSOR UNLIMITED, INC.
24,000 S. Peoria Ave.,
BEGGS, OK. 74421 **(918) 267-4961**
Prices shown above are for April 21, 1986
Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ $6.00, or Priority One @ $13.00!

Circle **no. 105** on reader service card.

Tree Shell

A Graphic Visual Shell for Unix/Xenix End-Users and Experts Alike!



Dealer inquiries welcomed.

COGITATE

"A Higher Form of Software"
24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345
TELEX: 386581 COGITATE USA

Circle **no. 81** on reader service card.

Dan Bricklin (VisiCalc)

"Sometimes the idea behind a program is one small creative effort. Just like in a short story, one little twist in the plot is the whole idea behind it."

Programmers at Work

Interviews with 19 of Today's Top Programmers
$14.95 (soft) **$19.95** (hard)
Available wherever fine books are sold.

MICROSOFT PRESS

Circle **no. 126** on reader service card.

Listing Two (Listing continued, text begins on page 38.)

```
        int     16H                     ; Scan the keyboard
        jz      Read_Keyboard_1         ; No character available
        mov     AH,0                    ; Yes
        int     16H                     ; Read keyboard
        cmp     AL,0                    ; Extended character
        je      Read_Keyboard_2         ; Yes
        mov     AH,0                    ; No, normal character
        ret

Read_Keyboard_1:
        mov     AX,-1                   ; Denote "no character available"
        ret

Read_Keyboard_2:                        ; Extended character
        mov     AL,AH
        mov     AH,01H                  ; Set the "function key" flags
        ret

Read_Keyboard endp

        endps
        end
```

End Listing Two

Listing Three

```
/*
 * This program emulates a dump terminal with file transfer support using
 * CompuServe's B-Protocol.  This program is just a sample of how to interface
 * the BP module (BP.C) with the rest of the terminal emulator.
 */

#define IBM_PC      1

extern int Transfer_File();  /* Transfer a file using the "B" protocol */

extern int Read_Keyboard();  /* Get a "raw" character from the keyboard */

extern Open_Modem();                /* Initialize the comm port */
extern int Read_Modem();     /* Read a character from the comm port */
```

(continued on page 102)

Time and Money.

We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

It's Expandable

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board...at your own pace, as your needs expand. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into **great** computers.

Battery Backup, Too

At 0.7 amps per 2 megabytes, SemiDisk consumes far less power than the competition. And you don't have to worry if the lights go out. The battery backup option gives you 5-10 hours of data protection during a blackout. Nobody else has this important feature. Why risk valuable data?

The Best News

	512K	1Mbyte	2Mbyte
SemiDisk I, S-100	$695	$1395	
SemiDisk II, S-100	$995		$1995
IBM PC, XT, AT	$595		$1795
QX-10	$595		$1795
TRS-80 II, 12, 16	$695		$1795
Battery Backup Unit	$150	$150	$150

SEMIDISK

Someday you'll get a SemiDisk.
Until then, you'll just have to....wait.

SemiDisk Systems, Inc., P.O. Box GG, Beaverton, Oregon 97075 503-642-3100

MasterCard VISA

Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards (300/1200 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle **no. 85** on reader service card.

XT new & AT HD TEST

HDTEST formats and tests hard drives in PC, XT, AT, and true compatibles. Written for production quality control; you know it's fast and thorough. Menu-driven operation and context sensitive help windows make life easy when installing or reformatting hard drives. Flag known bad tracks and do a comprehensive surface scan to find unlisted defects. You can modify controller and test setup, load/save setup and defect files. Works with WD, Xebec, OMTI, DTC, and Adaptec controllers. Free HD Drive Park Program included! Call for more information. $99.00

GALLERY 6 DISTRIBUTED BY PROTO PC, INC.
2439 FRANKLIN AVENUE ST. PAUL,MN 55114
612-644-4660 TLX910-380-7623

Circle **no. 295** on reader service card.

NOT COPY PROTECTED!

Sybil is an Advanced Diagnostics disk...
She can low format hard disks just like Advanced Diagnostics (IBM, Compaq, etc.) and she can do system and memory tests which provide even more information than Advanced Diagnostics does. $245.00 cheaper than IBM's Advanced Diagnostics!

Sybil is a Disaster Recovery program...
She can recover hard disks that have been accidentally formatted, completely! The hard disk reappears in exactly the same condition prior to the format. Truly amazing!

Sybil is a Graphics Editor...
She can draw on either RGB monitors (in color) or IBM Monochrome monitors in high ASCII characters. Perfect for creating Binary Image Files. The Binary Image Files can be converted to Assembly and then linked to other languages, such as your favorite Pascal, C, or compiled BASIC program. Includes source code.

Sybil is a File Wizard...
Sybil can backup files by **date**, by **time**, or by **size**. She can find any file (or files) anywhere on your hard or floppy disks, even if you haven't the vaguest notion. She can edit file attributes with the greatest of ease, unerase files, edit sectors, and globally change time and date stamps. All her file utilities understand paths and wildcards.

Sybil is also a...
RAM Disk, Print Spooler, General Regular Expression Parser and, **Advanced File Comparator**.

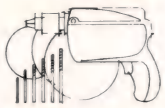
Order Sybil Today!
Call 800-922-3001, In Colorado, 303-444-1542

MasterCard VISA

SOPHCO
PO Box 7430
Boulder, Colorado 80306

AMERICAN EXPRESS

Circle **no. 298** on reader service card.



Power Tools for system builders™

Call today for our free catalog of design aids, compilers, libraries, debuggers, and support tools for Apple and IBM micro computers. The Power Tools catalog includes product descriptions, warranty and license terms, and all the information you need to make an intelligent purchase decision.

TSF offers technical support, competitive pricing, free UPS shipping on orders over $100, and a reasonable return policy. Visa, MasterCard, and American Express accepted without surcharge. **TSF helps you get your job done.**

Sample Prices:
Microsoft C $259
MASM 4.0 $109
Turbo Pascal $45
Mark Williams C $375
Lets C $59
Wendin OS Toolbox $89
Blaise Async Manager $137

Call Toll Free
24 hrs a day/7 days a week

Ask For Operator 2053

800-543-6277

Calif: 800-368-7600

TSF • Dept C-1 • 649 Mission Street • San Francisco • CA 94105 • (415) 957-0111
The Software Family™

Circle **no. 230** on reader service card.

Listing Three **(Listing continued, text begins on page 38.)**

```
extern int Write_Modem();     /* Send a character to the comm port */
extern Close_Modem();                 /* Release the comm port */

#define True  1
#define False  0

#define Baud_300    1         /* Baud rate codes used by Open_Modem */
#define Baud_450    2
#define Baud_1200   3
#define Baud_1800   4
#define Baud_2400   5
#define Baud_4800   6
#define Baud_9600   7

#ifdef IBM_PC                         /* for IBM style keyboards */
#define Exit_Key  0x012D      /* Alt-X */
#else
#define Exit_Key  0x001D      /* control-] */
#endif

#define Is_Function_Key(C)   ((C) > 127)

#define ENQ  0x05
#define DLE  0x10
#define ESC  0x1B

/*
 * We only support the B-protocol file transfer. No other VIDTEX features.
 */
static char VIDTEX_Response[] = "#DTE,PB,DT\015";

static int
    Old_Break_State,
    I,
    Ch,                               /* 16-bit "raw" character */
    Want_7_Bit,                       /* true if we want to ignore the parity bit */
    ESC_Seq_State;            /* Escape sequence state variable */


int Wants_To_Abort()
{
    return Read_Keyboard() == ESC;
}

main()
{
    char *cp;

    Want_7_Bit = True;
    ESC_Seq_State = 0;

#ifdef MSDOS
    Old_Break_State = Get_Break();
    Set_Break(0);
#endif

    Open_Modem(0, Baud_1200, False);
    puts("[ Terminal Mode ]");
    Ch = Read_Keyboard();

    while (Ch != Exit_Key)
          {
          if (Ch > 0)
              {
              if (Is_Function_Key(Ch))
                    {
                    /* Here to process any local function keys. */
                    }
              else
                    Write_Modem(Ch & 0x7F);
              }

          if ((Ch = Read_Modem()) >= 0)
              {
              if (Want_7_Bit) Ch &= 0x7F;

              switch (ESC_Seq_State)
                    {

                     case 0:
                         switch (Ch)
                               {
                               case ESC:
                                   ESC_Seq_State = 1;
                                   break;

                               case ENQ:
                                   /* Enquiry -- send ACK for packet 0 */

                                   Write_Modem(DLE);
                                   Write_Modem('0');
                                   break;

                               case DLE:
                                   ESC_Seq_State = 2;
                                   break;
```

```
                default:
                    Put_Char(Ch);
                }

            break;

        case 1:
            /* ESC -- process any escape sequences here */

            switch (Ch)
                {
                case 'I':
                    /*
                     * Reply to the VIDTEX "ESC I" identify sequence
                     */
                    cp = VIDTEX_Response;
                    while (*cp != 0) Write_Modem(*cp++);
                    ESC_Seq_State = 0;
                    break;

                default:
                    Put_Char(ESC);
                    Put_Char(Ch);
                    ESC_Seq_State = 0;
                }

            break;

        case 2:
            /* DLE */

            if (Ch == 'B')
                {
                /* Start of "B" protocol packet. Go into protocol
                 * mode and transfer the file as requested.
                 */

                if (!Transfer_File()) puts("Transfer failed!");
                }
            else
                {
                Put_Char(DLE);
                Put_Char(Ch);
                }

            ESC_Seq_State = 0;
        }
}
```

(continued on next page)

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

BTree Library — **75.00**

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

ISAM Driver — **40.00**

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

Make — **59.00**

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR — 149.00

For more information call or write:

softfocus

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

Credit cards accepted.

Dealer inquiries invited.

Circle **no. 259** on reader service card.

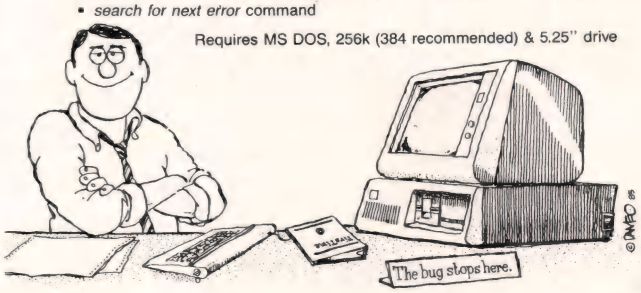
The First Idea-Processor For Programmers.

FirsTime™

Has features no other editor has.

- ☐ Fast program entry through single keystroke statement generators
- ☐ Fast editing through syntax oriented cursor movements
- ☐ Dramatically reduced debugging time through immediate syntax checking.
- ☐ The error checking is thorough and includes semantics • Undefined variables, types and constants • Assignment statements with mismatched types • Errors in include files and macro expansions
- ☐ Automatic program formatter (you specify the rules)
- ☐ Split Screen editing ☐ Command DOS from FirsTime
- ☐ Reading a file with errors moves cursor automatically to point of error
- ☐ Unique programmer-oriented features
  - *zoom* command gives top-down view of program logic
  - *view macro* command shows expansion of a C macro in the editor
  - *view/update* include file allows you to view and update an include file
  - *transform* command allows you to transform statements to related ones
  - *search for next error* command

Requires MS DOS, 256k (384 recommended) & 5.25" drive



To Order Call: (201) 741-8188 **or write:**

SPRUCE TECHNOLOGY CORPORATION

P.O. Box 7948
Shrewsbury, NJ 07701

FirsTime for Turbo Pascal	$ 74.95
FirsTime for dBase III	$125.00
FirsTime for MS-Pascal	$245.00
FirsTime for C	$295.00

FirsTime is a trademark of Spruce Technology Corporation • MS-DOS is a trademark of Microsoft Corporation • IBM is a trademark of International Business Machines Inc. • Turbo Pascal is a trademark of Borland International • dBase III is a trademark of Ashton-Tate.

Circle **no. 164** on reader service card.

Listing Three (Listing continued, text begins on page 38.)

```
            Ch = Read_Keyboard();
            }

        Close_Modem();

#ifdef MSDOS
    Set_Break(Old_Break_State);
#endif
}
```

End Listing Three

Listing Four

```
            title     Screen
            include   \lc\dos.mac

Video               equ       10H               ; IBM BIOS call
TTY_Write equ       14

            pseg

            public    Put_Char

Put_Char proc
;++
; Function:
;           Write a character to the screen in "normal" TTY-style output.
;
; Inputs:
;           4[BP]     - the character to write
;
; Outputs: none
;
; Returns: nothing
;--
            push      BP
            mov       BP,SP
            mov       AL,4[BP]          ; Character to write
            mov       BH,0                        ; Current page
            mov       AH,TTY_Write
            int       Video
            pop       BP
            ret
Put_Char endp

            endps
            end
```

End Listing Four

(Listings to be continued next month.)

TRUE MULTI-TASKING!

TASKVIEW is high tech, available now, and it works with virtually all DOS software. Give Lotus, Sidekick, Multimate or most any DOS program the advantages of real multi-tasking. It's simple to use, compatible, bulletproof and most of all, it won't slow you down. That's because TASKVIEW only shares your computer when YOU want it shared. At other times, your visible program runs at full speed, waiting for you to easily switch from program to program at the touch of a key. Compatible with most DOS computers including the IBM PC/XT/AT/Jr. series, you can order TASKVIEW today for only $69.95 + 5.00 Shipping and Handling, VISA and Mastercard.

ORDER LINE
(206) 367-0650

30 Day Money Back Guarantee.

Taskview trademark Sunny Hill Software.
Lotus trademark Lotus Development Corp.
Sidekick trademark Borland Intl.
Multimate trademark Ashton Tate.

Sunny Hill Software
13732 Midvale North Suite 206
Seattle, Washington 98133

Circle **no. 172** on reader service card.

DR. JACK PURDUM

Get the proven product from the man who wrote the books on "C".

C Programming Guide
After reading the 1st edition, Jerry Pournelle (**BYTE** Magazine) said: "I recommend this book...Read it *before* trying to tackle Kernighan and Ritchie." The second edition expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.

$20

C Self-Study Guide
(Purdum, Que Corp.) Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.

$17

C Programmer's Library
(Purdum, Leslie, Stegemoller, Que Corp.) This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

$22

CED Program Editor
CED now supports on-line function help. If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look-alikes.

$29.95

C Source for Standard Library
Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendentals and functions written in assembler.

$10 ($20 if not with order)

Developer's Library
Contains all the source code for the standard library, including transcendental and assembler functions. Available with compiler purchase only.

$25 ($50 if not with order)

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

THE FIRST PROFESSIONAL "C" COMPILER FOR UNDER $60.00

Limited Time Offer
FREE
CED TEXT EDITOR WITH "BUILT-IN" FUNCTION HELP.

NOTHING IN THIS PRICE RANGE EVEN COMES CLOSE.

"Ecosoft's Eco-C is going to turn some major heads."

"Eco-C is the first compiler reviewed that has clearly begun implementing the coming ANSI standard. Eco-C supports prototyping."

"Eco-C performed well on all the benchmarks, generating code that was quite comparable to that of compilers 10 times as costly."

from:
Christopher Skelly
Computer Language,
Feb., 1986

"The driver program is another strength: it compiles and links a list of files and provides a simple MAKE capability."

"This compiler does handle syntax errors much better than average—no avalanche of spurious messages here."

from:
William Hunt
PC Tech Journal,
Jan., 1986

"Eco-C88 is a high-quality package... comparable to systems costing much more."

"Eco-C88 is one of the fastest..."

"It is convenient to use, works well, and produces acceptably compact and fast programs."

from:
Dr. David Clark
Byte, Jan., 1986

Minimum System Requirements:

To use Eco-C88 Release 3.0, you must have:

1. An IBM PC, XT, or AT-compatible computer with monitor.
2. 256K or more memory.
3. Two 360K disk drives, or a hard disk.
4. PC or MSDOS 2.1 or later to include the MSDOS linker.

Eco-C88, mini-make, memfiles and CED are trademarks of Ecosoft Inc.
IBM is a trademark of International Business Machines.
UNIX is a trademark of Bell Labs.
MSDOS and MASM are trademarks of Microsoft.

Full-featured C compiler. Supports all C features, data types (except bit fields), and operators.

New Language Enhancements. You also get prototyping, enum and void data types, plus structure passing and assignment.

Tiered Error Checking. All syntax errors are automatically flagged, but you can select the level of "link-like" semantic error checking you want.

Complete Standard Library. Over 200 functions, many of which are System V compatible for greater source code portability.

Screen and Memory Functions. Now you can write programs that use color, cursor addressing, even ones that let you design your own graphics functions. You also get memfiles™ that allow you to access memory outside the normal data segment as a file.

8087 and 80287 Support. If you have one of these math chips, your programs will take immediate advantage of it. If you don't have one, the code automatically switches to software floating point.

Full Screen Editor. The CED editor is a full screen editor with multiple windows, macro commands, on-line function help, plus a full set of editing commands. (Requires a true IBM PC compatible.) You can edit, compile, link, and execute programs from the editor which greatly reduces development time.

Includes a cc and mini-make™. The UNIX-like cc makes compiling programs a snap. You can run cc from within the CED editor.

ASM or OBJ Output. You can select assembler or relocatable output from the compiler. Both are MASM compatible and ready for use with the MSDOS linker to produce EXE files.

	Eco-C88	Lattice	Computer Inn. C86	Microsoft	Mark Williams
sieve	12	11	13	11	12
fib	43	58	46	109	–
deref	14	13	–	10	11
matrix	22	29	27	28	29

Computer Language, Feb., 1985, p. 79. Reproduced with permission.

Orders only: 1-800-952-0472

ORDER FORM **CLIP & MAIL TO:** Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

- ☐ C Compiler $59.95 _____
- ☐ C Programming Guide $20.00 _____
- ☐ C Self-Study Guide $17.00 _____
- ☐ C Programmer's Library $22.00 _____
- ☐ CED Program Editor $29.95 _____
- ☐ C Source for Standard Library $10.00 ($20.00 if not with order) _____
- ☐ Developer's Library $25.00 ($50.00 if not with order) _____

SHIPPING $4.00 _____
TOTAL (IND. RES. ADD 5% TAX) _____

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK
CARD # _____ EXPIR. DATE _____
NAME _____
ADDRESS _____
CITY _____ STATE _____
ZIP _____ PHONE _____

ECOSOFT

Circle no. 89 on reader service card

FORTRAN PROGRAMMERS

Downloading from mainframes or developing on the PC the choice is F77L.

"Lahey's F77L FORTRAN is the compiler of choice... F77L compiled the five files in a total of 12 minutes which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN." PC Magazine

"The manual that comes with this compiler is well put together. The messages are clearly explained, the compiler's unique features are well documented... All in all, F77L is a fine, well supported product that we think will do very well in the marketplace."

Computer Language

VERSION 2.0 NOW AVAILABLE — $477

Full ANSI FORTRAN-77
Source On-Line Debugger
Common/Array greater than 64K
Lattice C and other 3rd Party Compatibility

To order or for more information

(213) 541-1200

Lahey Computer Systems, Inc.
31244 Palos Verdes Drive West, Suite #243
Rancho Palos Verdes, CA 90274

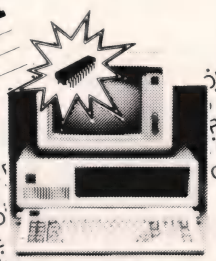
Requires MS-DOS and 8087

MS-DOS and MS FORTRAN are trademarks of Microsoft Corporation
Pro FORTRAN is a trademark of International Business Machines

Circle **no. 186** on reader service card.

PUT A CP/M COMPUTER IN YOUR PC!

RUN/CPM™



ONLY $99.95
INCLUDING HARDWARE!
Plus over 20 CP/M80 programs!

And run 1,000's of CP/M programs
up to 30% faster, directly from your CP/M disks!

How does it work?
RUN/CPM virtually transforms your PC into any of the most popular CP/M systems. A simple replacement of your PC's 8088/86 microprocessor with our N.E.C. V-20/30 microprocessor gives your computer the ability to run both 8 bit CP/M and 16 bit MS-DOS programs. RUN/CPM will transform your PC's floppy drives into CP/M drives able to directly read, write, and format over 100 CP/M disks! Terminal emulation supporting dozens of the most popular terminals completes the transformation of your PC into a CP/M system.

Performance?
Depending on the application, many of your CP/M programs will run up to 30% faster on your PC. Other features include; ability to run CP/M programs in color, logical and physical drive assignments, run CP/M or MS-DOS programs from the same prompt. RUN/CPM is the solution to running CP/M software on PC's.

To order
send check or money
order (U.S. funds)
add $5.00 shipping/handling.

Micro Interfaces Corporation
6824 N.W. 169th Street, Miami, Florida 33015
(305) 823-8088
Telex 5106004680 MICRO INTER CO
Ask About Our Intel Operating System Interfaces

OEM, VAR, Dealers, Inquiries Invited

ORDERS ONLY

1-800-637-7226

Trademarks: CP/M (Digital Research Inc.); IBM (IBM Corp.); RUN/CPM (Micro Interfaces Corp.); MS-DOS (Microsoft Inc.); INTEL (Intel Corp.)

Circle **no. 110** on reader service card.

16-BIT

Listing One (Text begins on page 112.)

```
        mov     dx,seg fname    ; open a file for I/O
        mov     ds,dx           ; DS:DX = ASCIIZ filespec
        mov     dx,offset fname ; function 3DH = open,
        mov     ax,3d02h        ; use mode 2 (read/write)
        int     21h
        jc      error           ; jump if open failed
        mov     my_file,ax      ; save handle for file

        .
        .                       ; further file
        .                       ; processing here
        .

                                ; now use DUP and CLOSE
                                ; to update the directory...
        mov     bx,my_file      ; get handle for file
        mov     ah,45h          ; function 45H = DUP handle,
        int     21h
        jc      error           ; jump if DUP failed
        mov     bx,ax           ; now close the DUP'd handle
        mov     ah,3eh          ; function 3EH = close file
        int     21h             ; transfer to MS-DOS
        jc      error           ; jump if close failed
                                ; otherwise directory is
                                ; updated, continue processing
        .
        .
error:  .
        .
        .

my_file dw      0               ; handle from previous "open"

fname   db      'MYFILE.DAT',0  ; ASCIIZ filespec
```

End Listing One

Listing Two

```
page 60, 120
title Redirected I/O example

;
; REDIRECT.ASM --- An illustration of
; I/O redirection under MS-DOS 2.x or 3.x
;
; Copyright (C) 1985 by Jerry Jankura
; Created:   6 November 1985
; Modified:  9 November 1985
;
; Abstract: This routine demonstrates redirection of I/O
; from the console to a line printer.  The method may be
; used to redirect I/O from any device to any other device.
; Requires Microsoft MS-DOS 2.X or 3.X, or DRI Concurrent
; DOS version 4.1.
;

STD_IN          EQU     0       ; Standard input handle
STD_OUT         EQU     1       ; Standard output handle
STD_ERR         EQU     2       ; Standare error handle
STD_AUX         EQU     3       ; Standard Auxiliary handle
STD_LST         EQU     4       ; Standard printer handle

C_WRITESTR      EQU     9       ; Write string to STD_OUT

F_DUP           EQU     45H     ; Duplicate handle
F_CDUP          EQU     46H     ; Force duplicate handle
F_CLOSE         EQU     3EH     ; Close file handle
F_WRITE         EQU     40H     ; Write to file or dev

P_TERM          EQU     4CH     ; Terminate a program

MS_DOS          EQU     21H     ; MS-DOS service request

CR              EQU     0DH     ; Carriage return
LF              EQU     0AH     ; Line feed

datasg          segment para    'data'


msg1            db      CR, LF, 'Redirected I/O example....'
                db      CR, LF
                db      CR, LF, 'This example was written using the'
                db      CR, LF, 'File I/O system services, with the'
                db      CR, LF, 'file handle being set to STD_OUT.'
                db      CR, LF, 'STD_OUT normally defaults to the'
                db      CR, LF, 'video screen, so you are reading'
                db      CR, LF, 'this message on the screen.'
                db      CR, LF

msg2            db      CR, LF, 'However, we may direct STD_OUT to'
                db      CR, LF, 'another device, such as the printer.'
                db      CR, LF, 'This message is still written to'
                db      CR, LF, 'STD-OUT, but is appears at the printer.'
                db      CR, LF, 'Again, the operating system provides'
                db      CR, LF, 'the facility to allow one file to mimic'
                db      CR, LF, 'and track another.  The Command processor'
                db      CR, LF, 'normally implements this redirection'
                db      CR, LF, 'of standard devices.'
                db      CR, LF

msg3            db      CR, LF, 'This message is written on the'
                db      CR, LF, 'Video screen, demonstrating that'
                db      CR, LF, 'a message may be redirected to the'
                db      CR, LF, 'normal STD_OUT device in the same'
                db      CR, LF, 'manner that was used to redirect'
                db      CR, LF, 'it to the printer.'
                db      CR, LF
```

```
                db      CR, LF, 'Note also that the initialized data is'
                db      CR, LF, 'stored in the data segment, rather'
                db      CR, LF, 'than in the code segment.'
                db      CR, LF
                db      CR, LF, 'Also, the messages are written using'
                db      CR, LF, 'block I/O, so a minimum number of DOS'
                db      CR, LF, 'system services are requested.'

msg4            db      0

dup_handle      dw      ?
orig_handle     dw      ?

datasg          ends

stacksg         segment para stack 'stack'

mystack         db      512 dup (?)

stacksg         ends

code            segment para 'code'

assume          cs: code
assume          ds: datasg
assume          ss: stacksg
assume          es: nothing

test_redirect   proc    far

; Initialize stack pointer and data segment register
; to the correct values.  The stack pointer is set
; to the top of the stack segment.  The data segment
; is set to the segment of the first variable.  Note
; that at this point in time, the DS register does
; not point to the PSP.

                mov     sp, 513 ; set up user stack
                mov     ax, seg msg1
                mov     ds, ax

; First, write a sign-on message to the screen.  We
; will attempt to write this message to the standard
; output device.

;       ah:     INT 21H function id.
;       bx:     file handle
;       cx:     # of bytes to transfer
;       DS:dx   points to message

                mov     ah, F_WRITE
                mov     dx, offset msg1
                mov     bx, STD_OUT
                mov     cx, msg2-msg1
                int     MS_DOS

; Now, we wish to redirect the output to the
; printer.  Before we force the redirection,
; we must make a copy of the standard output
; file handle and store it in the field
; orig_handle.

                mov     bx, STD_OUT
                mov     ah, F_DUP
                int     MS_DOS
                mov     word ptr orig_handle, ax
                mov     bx, STD_LST
                mov     ah, F_DUP
                int     MS_DOS
                mov     word ptr dup_handle, ax

; Then, the STD_LST handle is set to track
; the STD_OUT file.

                mov     bx, ax
                mov     cx, STD_OUT
                mov     ah, F_CDUP
                int     MS_DOS

; Let's write a message out and try it.
; Note that we are still writing information to
; the STD_OUT device.

                mov     ah, F_WRITE
                mov     bx, STD_OUT
                mov     cx, msg3-msg2
                mov     dx, offset msg2
                int     MS_DOS

; Now, let's clean up and return everything
; back to its original condition.

                mov     bx, word ptr dup_handle
                mov     ah, F_CLOSE
                int     MS_DOS

                mov     bx, word ptr orig_handle
                mov     cx, STD_OUT
                mov     ah, F_CDUP
                int     MS_DOS

                mov     ah, F_WRITE
                mov     bx, STD_OUT
                mov     cx, msg4-msg3
                mov     dx, offset msg3
                int     MS_DOS

                mov     ah, P_TERM
                int     MS_DOS

test_redirect   endp

code            ends

                end     test_redirect
```

End Listings

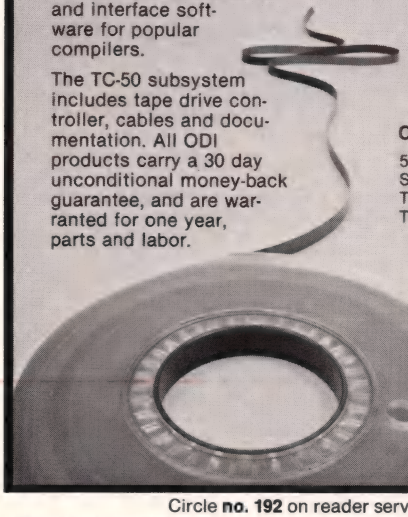
nine track tape users

Micro to Mainframe Connection

The Model TC-50 ½-inch tape subsystem provides a standard medium for transmission of mainframe data base information to PC users, while maintaining mainframe isolation and data integrity. Use ODI subsystems to import data to data base programs like dBase III.

The TC-50 subsystem also provides fast back-up capability as well as a device driver and interface software for popular compilers.

The TC-50 subsystem includes tape drive controller, cables and documentation. All ODI products carry a 30 day unconditional money-back guarantee, and are warranted for one year, parts and labor.



ODI

Overland Data, Inc.

5644 Kearny Mesa Road
San Diego, CA 92111
Tel. (619) 571-5555
Telex 754923 OVERLAND

Also Available – XENIX tape subsystems for the IBM AT

Circle **no. 192** on reader service card.

MODULA-2

DEVELOPMENT BUILDING BLOCKS

REPERTOIRE, from PMI. High-performance tools. Screen display system. Multi-window editor. Full source, so your programs can follow when you change machines.

Screen System.

REPERTOIRE won't bloat your programs because it doesn't generate code. Create screens exactly as they will look, then compress them into one dense, rapid-access file. REPERTOIRE lets your program display a screen instantly in any window with a single function call. Screens check user input, scroll within windows, give context-sensitive help, and conditionally branch to other screens using natural-language analysis functions that you imbed in the screens.

Editor.

REPERTOIRE's full-screen multi-window editor fits neatly into your programs. It lets your users create and edit multiple files concurrently.

High-Performance Low level Routines.

REPERTOIRE provides improved DOS and BIOS access, speaker control, string tools, list handling, and a sample directory manager.

Excellent for educational software or any other screen-intensive application. For IBM compatibles. Manual and two 360K diskettes. Logitech & ITC versions, $64 each. Both versions for $84. Check/MC/VISA. **Send for FREE documentation and demo disk to find out more.**

PMI 4536 S.E. 50th Portland, OR 97206 (503) 293-7706
MCI Mail: PMI; Compuserve: 74706,262

Circle **no. 239** on reader service card.

Listing One (Text begins on page 116.)

```
{------------------ Constants and Data Types Needed ----------}
CONST MAX_SWITCH = 3;
      MAX_ELEMENTS = 100;
      MAX_BIN = 30;
      MAX_BIN_PLUS_ONE = 31;

TYPE Histogram_Rec =

     RECORD
        Num_Elements : 1..MAX_ELEMENTS;
        Switch : 1..MAX_SWITCH;
        Num_Bins : 1..MAX_BIN_PLUS_ONE;
        Count : ARRAY [1..MAX_BIN] OF INTEGER;  { For output only }
        CASE INTEGER OF
          1 : (Real_Array : ARRAY [1..MAX_ELEMENTS] OF REAL;
               Real_Bins  : ARRAY [1..MAX_BIN_PLUS_ONE] OF REAL);
          2 : (String_Array : ARRAY [1..MAX_ELEMENTS] OF STRING[80];
               First_Char, Last_Char : INTEGER;
               String_Bins  : ARRAY [1..MAX_BIN_PLUS_ONE] OF STRING[20]);
{ ---------------- More types here ------------------------------}
{       3 : (Intg_Array : ARRAY [1..MAX_ELEMENTS] OF INTEGER;       }
{            Intg_Bins  : ARRAY [1..MAX_BIN_PLUS_ONE] OF INTEGER);  }
{       4 : (Char_Array : ARRAY [1..MAX_ELEMENTS] OF CHAR;          }
{            Char_Bins  : ARRAY [1..MAX_BIN_PLUS_ONE] OF CHAR);     }

     END;


PROCEDURE Count_Histogram(VAR Histogram : Histogram_Rec);
{ Pseudo-overloaded histogram counting procedure }

VAR I, J :INTEGER;
    Found : BOOLEAN;

PROCEDURE Real_Histogram;
{ Local procedure to count histogram frequency for an array of reals }

BEGIN
    WITH Histogram DO BEGIN
        FOR I := 1 TO Num_Elements DO BEGIN { main loop }
            { Is element within bin ranges ? }
            IF (Real_Array[I] >= Real_Bins[1]) AND
               (Real_Array[I] < Real_Bins[Num_Bins])
            THEN BEGIN  { Locate corresponding bin }
                J := 1; Found := FALSE;
                WHILE (J < Num_Bins) AND (NOT Found) DO
                    IF (Real_Array[I] >= Real_Bins[J]) AND
                       (Real_Array[I] < Real_Bins[J+1])
                       THEN Found := TRUE
                       ELSE J := J + 1;
                { END WHILE }
                Count[J] := Count[J] + 1;
            END; { IF }
        END; { FOR I }
    END; { WITH }
END; { Real_Histogram }


PROCEDURE String_Histogram;
{ Procedure to count histogram frequency for an array of strings }

VAR Strr  : STRING[20];
    Copy_String : STRING[80];

BEGIN
    WITH Histogram DO BEGIN
        FOR I := 1 TO Num_Elements DO BEGIN  { main loop }
            Copy_String := String_Array[I];
            Strr := ''; { initialize Strr }
            { Extract portion of string for comparison }
            FOR J := First_Char TO Last_Char DO
                Strr := Strr + Copy_String[J];

            { Is element within bin ranges ? }
            IF (Strr >= String_Bins[1]) AND
               (Strr < String_Bins[Num_Bins])
            THEN BEGIN
                J := 1; Found := FALSE;
                WHILE (J < Num_Bins) AND (NOT Found) DO
                    IF (Strr >= String_Bins[J]) AND
                       (Strr < String_Bins[J+1])
                       THEN Found := TRUE
                       ELSE J := J + 1;
                { END WHILE }
                Count[J] := Count[J] + 1;
            END; { IF }
        END; { FOR I }
    END; { WITH }

END; { String_Histogram }

BEGIN
    { Initialize keys }
    FOR I := 1 TO MAX_BIN DO
        Histogram.Count[I] := 0;

    CASE Histogram.Switch OF
        1 : Real_Histogram;  { Do histogram count for reals }
        2 : String_Histogram; { Do histogram count for strings }
    END; { CASE }
END; {  Count_Histogram }
```

End Listing One

Listing Two

```
{------------------ Constants and Data Types Needed ----------}
TYPE Complex = RECORD
                 Is_Polar : BOOLEAN;
                 CASE BOOLEAN OF
                     { Polar coordinates }
                      TRUE  : (Modulus, Angle : REAL);
                      { Rectangular coordinates }
                      FALSE : (Xcoord, YCoord : REAL);
               END;


PROCEDURE Add(A, B : Complex; { input }
              VAR C : Complex { ouput });
{ Procedure to add two complex numbers taking into account }
{ their dual presentation.          }
{ local rectangular coordinates }
VAR X1, X2, X3, Y1, Y2, Y3 : REAL;

PROCEDURE Get_Coordinates(P : Complex; { input }
                          X, Y : REAL { output });
{ Local procedure to obtain rectangular coordinates }
BEGIN
 WITH P DO BEGIN
    IF P.Is_Polar
    THEN BEGIN
        X := Modulus * COS(Angle);
        Y := Modulus * SIN(Angle)
    END
    ELSE BEGIN
        X := Xcoord;
        Y := Ycoord
    END; { IF }
 END; { WITH }
END; { Get_Coordinates }

BEGIN
    { Get rectangular coordinates of A and B }
    Get_Coordinates(A, X1, Y1);
    Get_Coordinates(B, X2, Y2);

    { Add rectangular components }
    X3 := X1 + X2; Y3 := Y1 + Y2;

    WITH C DO BEGIN
        IF Is_Polar
        THEN BEGIN
            Modulus := SQRT(X3*X3 + Y3*Y3);
            Angle := ArcTan(Y3/X3)
        END
        ELSE BEGIN
            Xcoord := X3;
            Ycoord := Y3
        END; { IF }
    END; { WITH }
END; { Add }
```

End Listing Two

Listing Three

```
{-------------- Constants and Data Types Needed --------------}
CONST MAX_HEIGHT = 100;

TYPE

     Complex = RECORD Reel, Imaginary : REAL; END;

     Stack_Rec =
        RECORD
            Switch : INTEGER;
            CASE INTEGER OF
                0 : (Integer_type : INTEGER);
                1 : (Real_type : REAL);
                2 : (String_type : STRING[80]);
                3 : (Complex_type : Complex);
        END;

    Stack = RECORD
                Height : INTEGER;
                Stack_Member : ARRAY [1..MAX_HEIGHT] OF Stack_Rec;
            END;

PROCEDURE Push(VAR Stk : Stack;    { in/out }
               Element : Stack_Rec; { output }
               VAR OK : BOOLEAN    { output });
{ Procedure to push 'Element' in stack }
BEGIN
    WITH Stk DO BEGIN
        OK := FALSE;
        IF Height < MAX_HEIGHT
        THEN BEGIN
            OK := TRUE;
            Height := Height + 1;
            Stack_Member[Height] := Element
        END; { IF }
    END; { WITH }
END; { Push }


PROCEDURE Pop(VAR Stk : Stack;    { in/out }
              VAR Element : Stack_Rec; { output }
              VAR OK : BOOLEAN    { output });
{ Procedure to pop 'Element' in stack }
BEGIN
    WITH Stk DO BEGIN
        OK := FALSE;
        IF Height > 0
```

```
        THEN BEGIN
            OK := TRUE;
            Element := Stack_Member[Height];
            Height := Height - 1
        END; { IF }
    END; { WITH }
END; { Push }


PROCEDURE Selective_Pop(VAR Stk : Stack;   { in/out }
                        VAR Element : Stack_Rec; { in/out }
                        VAR OK : BOOLEAN   { output });
{ Procedure to search for first stack element that matches }
{ the Switch field in 'Element'.          }

VAR I, J : INTEGER;

BEGIN
    WITH Stk DO BEGIN
        OK := FALSE;
        I := Height;
        { Attempt to locate element of desired type }
        WHILE (I > 0) AND (NOT OK) DO
            IF Element.Switch = Stack_Member[I].Switch
            THEN OK := TRUE
            ELSE I := I - 1;

        IF OK THEN BEGIN { Found one! }
            Element := Stack_Member[I];
            { Rearrange stack }
            FOR J := I TO Height-1 DO
                Stack_Member[J] := Stack_Member[J+1];
            Height := Height - 1;
        END; { IF }
    END; { WITH }
END; { Selective_Pop }
```

End Listing Three

Listing Four A

```
DEFINITION MODULE HPStackMod;

EXPORT QUALIFIED
    HPStack, (* Opaque type *)
    Enter, Clst, Add, Sub, Mul, Div, RclLast, GetX; (* Procedures *)

TYPE HPStack;

PROCEDURE Enter(VAR Stack : HPStack; (* in/out *)
                X : REAL             (* input  *));
(* Procedure to enter a number in the stack *)

PROCEDURE Clst(VAR Stack : HPStack (* in/out *));
(* Procedure to clear stack and LASTX register *)

PROCEDURE Add(VAR Stack : HPStack (* in/out *));
(* Procedure to add Y and X registers *)

PROCEDURE Sub(VAR Stack : HPStack (* in/out *));
(* Procedure to subtract Y and X registers *)
```

End Listing Four A

Listing Four B

```
PROCEDURE Mul(VAR Stack : HPStack (* in/out *));
(* Procedure to multiply Y and X registers *)

PROCEDURE Div(VAR Stack : HPStack;  (* in/out *)
              VAR OK : BOOLEAN      (* output *));
(* Procedure to divide Y and X registers *)

PROCEDURE RclLast(VAR Stack : HPStack (* in/out *));
(* Procedure to recall LASTX register *)

PROCEDURE GetX(Stack : HPStack (* input *)) : REAL;
(* Function to get X register *)

END HPStackMod.


Listing Four B.
IMPLEMENTATION MODULE HPStackMod;
(* Module implementing scalar-based RPN stack calculator *)

TYPE HPStackRec = RECORD
                    XReg, YReg, ZReg, TReg, LASTX : REAL;
                  END;
    (* Exported opaque type *)
    HPStack = POINTER TO HPStackRec;

PROCEDURE StackDown;
(*------- Internal module usage ------*)
(* Procedure to roll down Y, Z and T registers *)
BEGIN
    YReg := ZReg;  (* Copy Z into Y *)
    ZReg := TReg   (* Copy T into Z *)
END StackDown;

PROCEDURE Enter(VAR Stack : HPStack; (* in/out *)
                X : REAL             (* input  *));
(* Procedure to enter a number in the stack and push it *)

BEGIN
    WITH Stack^ DO
        TReg := ZReg; ZReg := YReg;
        YReg := XReg; XReg:= X
    END;
END Enter;
```

(continued on next page)

CACHE22 + CP/M 2.2 = CP/M Max!

CACHE22 is a front-end system program that buries all of CP/M 2.2 in banked memory. It helps 8080/Z80 computers to survive by providing up to 63.25K of TPA, plus the ability to speed disk operations, eliminate system tracks, and run Sidekick-style software without loss of transient program space. Complete source and installation manual, $50.00.

CP/M is a trademark of Digital Research Inc.
Sidekick is a trademark of Borland International

MIKEN OPTICAL COMPANY
53 Abbett Avenue, Morristown, NJ 07960
(201) 267-1210

Circle **no. 261** on reader service card.

WORLD'S FIRST ASSEMBLER INTERPRETER

Advanced Trace86™ 2.0

- Create .COM programs with BASIC-like commands: LOAD, SAVE, EDIT, LIST, RUN. Reloadable into AT86, complete with all labels, comments & variable names.
- Macro Assembler support-scan .MAP and .LST files for labels and variable names
- Full-screen symbolic tracing of any program.
- Powerful conditional breakpoint facilities, including hardware "button" support (if installed)
- "Screen save" option, or use dual monitors
- Hex/decimal calculator & converter
- Best 8087/80286/80287 support on the market!
- And more . . . Priced at $175.00

To order or request more information contact:

Morgan Computing Co., Inc.
(214) 245-4763
P.O. Box 112730, Carrollton, TX 75011

Circle **no. 128** on reader service card.

Listing Four B (Listing continued, text begins on page 116.)

```
PROCEDURE Clst(VAR Stack : HPStack (* in/out *));
(* Procedure to clear stack and LASTX register *)

BEGIN
    WITH Stack^ DO
        XReg := 0.0; YReg := 0.0; ZReg := 0.0;
        TReg := 0.0; LASTX := 0.0;
    END;
END Clst;

PROCEDURE Add(VAR Stack : HPStack (* in/out *));
(* Procedure to add Y and X registers *)

BEGIN
    WITH Stack^ DO
        LASTX := XReg; (* Save X reg. in LASTX *)
        XReg := YReg + XReg;
        StackDown
    END;
END Add;

PROCEDURE Sub(VAR Stack : HPStack (* in/out *));
(* Procedure to subtract Y and X registers *)

BEGIN
    WITH Stack^ DO
        LASTX := XReg; (* Save X reg. in LASTX *)
        XReg := YReg - XReg;
        StackDown
    END;
END Sub;

PROCEDURE Mul(VAR Stack : HPStack (* in/out *));
(* Procedure to multiply Y and X registers *)

BEGIN
    WITH Stack^ DO
        LASTX := XReg; (* Save X reg. in LASTX *)
        XReg := YReg * XReg;
        StackDown
    END;
END Mul;

PROCEDURE Div(VAR Stack : HPStack;  (* in/out *)
              VAR OK : BOOLEAN      (* output *));
(* Procedure to divide Y and X registers *)
```

End Listing Four B

Listing Four C

```
BEGIN
    OK := TRUE;
    WITH Stack^ DO
        IF StackReg[1] <> 0.0 (* Division by non-zero ? *)
        THEN
            LASTX := XReg; (* Save X reg. in LASTX *)
            XReg := YReg / XReg;
            StackDown
        ELSE (* Trouble *)
            OK := FALSE
        END;
    END;
END Div;

PROCEDURE RclLast(VAR Stack : HPStack (* in/out *));
(* Procedure to recall LASTX register *)

BEGIN
    WITH Stack^ DO
        TReg := ZReg; ZReg := YReg;
        YReg := XReg; XReg := LASTX
    END;
END RclLast;

PROCEDURE GetX(Stack : HPStack (* input *)) : REAL;
(* Function to get X register *)
BEGIN
    RETURN Stack^.XReg;
END GetX;

END HPStackMod.

Listing Four C.


IMPLEMENTATION MODULE HPStackMod;
(* Module implementing array-based RPN stack calculator *)


TYPE HPStackRec = RECORD
                    StackReg : ARRAY [0..4] OF REAL;
          (* StackReg[0] is LASTX, StackReg[1] is X Reg *)
          (* StackReg[2] is Y Reg, StackReg[3] is Z Reg *)
          (* StackReg[4] is T Reg *)
                  END;

    (* Exported opaque type *)
    HPStack = POINTER TO HPStackRec;

PROCEDURE StackDown;
(*------- Internal module usage -----*)
(* Procedure to roll down Y, Z and T registers *)
BEGIN
    StackReg[2] := StackReg[3];  (* Copy Z into Y *)
    StackReg[3] := StackReg[4] (* Copy T into Z *)
END StackDown;
```

```
PROCEDURE Enter(VAR Stack : HPStack; (* in/out *)
                X : REAL             (* input *));
(* Procedure to enter a number in the stack *)

VAR I : CARDINAL;

BEGIN
    WITH Stack^ DO
        FOR I := 3 TO 1 BY -1 DO
            StackReg[I+1] := StackReg[I]
        END;
        StackReg[1] := X
    END;
END Enter;

PROCEDURE Clst(VAR Stack : HPStack (* in/out *));
(* Procedure to clear stack and LASTX register *)
VAR I : CARDINAL;
BEGIN
   WITH Stack^ DO
       FOR I := 0 TO 4 DO
           StackReg[I] := 0.0
       END;
   END;
END Clst;

PROCEDURE Add(VAR Stack : HPStack (* in/out *));
(* Procedure to add Y and X registers *)

BEGIN
    WITH Stack^ DO
        StackReg[0] := StackReg[1]; (* Save X reg. in LASTX *)
        StackReg[1] := StackReg[2] + StackReg[1];
        StackDown

    END;
END Add;

PROCEDURE Sub(VAR Stack : HPStack  (* in/out *));
(* Procedure to subtract Y and X registers *)

BEGIN
    WITH Stack^ DO
        StackReg[0] := StackReg[1]; (* Save X reg. in LASTX *)
        StackReg[1] := StackReg[2] - StackReg[1];
        StackDown
    END;
END Sub;

PROCEDURE Mul(VAR Stack : HPStack  (* in/out *));
(* Procedure to multiply Y and X registers *)

BEGIN
    WITH Stack^ DO
        StackReg[0] := StackReg[1]; (* Save X reg. in LASTX *)
        StackReg[1] := StackReg[2] * StackReg[1];
        StackDown
    END;
END Mul;

PROCEDURE Div(VAR Stack : HPStack;  (* in/out *)
              VAR OK : BOOLEAN      (* output *));
(* Procedure to divide Y and X registers *)

BEGIN
    OK := TRUE;
    WITH Stack^ DO
        IF StackReg[1] <> 0.0 (* Division by non-zero ? *)
        THEN
            StackReg[0] := StackReg[1]; (* Save X reg. in LASTX *)
            StackReg[1] := StackReg[2] / StackReg[1];
            StackDown
        ELSE (* Trouble *)
            OK := FALSE
        END;
    END;
END Div;

PROCEDURE RclLast(VAR Stack : HPStack  (* in/out *));
(* Procedure to recall LASTX register *)
VAR I : CARDINAL;
BEGIN
    WITH Stack^ DO
        FOR I := 4 TO 1 BY -1 DO
            StackReg[I] := StackReg[I-1]
        END;
    END;
END RclLast;

PROCEDURE GetX(Stack : HPStack (* input *)) : REAL;
(* Function to get X register *)
BEGIN
    RETURN Stack^.StackReg[1];
END GetX;

END HPStackMod.
```

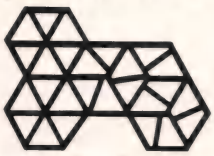
End Listings

BYSO™ LISP

A production LISP compiler designed for optimum performance on the IBM PC.

- Includes interpreter — debug programs instantly.
- Compatible with Common LISP and some earlier dialects.
- Access to all system functions of IBM PC.
- Complete — has full screen editor, only PCDOS required.
- Tested and reliable — in use since Aug. '84, all known bugs fixed.
- Very fast — does Jul. '84 BYTE Sieve test in .8 seconds. Other LISPs take 30 to 80 seconds. Garbage collects in under a quarter second.
- Visual Syntax™ — an editor written in BYSO LISP that displays programs graphically.
- Fully documented — all features explained, application notes given, full source code of Visual Syntax.
- Not copy protected, preinstalled.
- Generates stand alone code that is royalty free in most cases.
- Priced as a good C compiler, not as a miracle expert system.

Interpreter only, single user license $150.
Compiler and interpreter, single user license $395.
Requires 256K, IBM PC or true compatible.



LEVIEN INSTRUMENT CO.
Sittlington Hill/P.O. Box 31
McDowell, VA 24458
(703) 396-3345

BYSO is a trademark of Raphael L. Levien. IBM PC is a registered trademark of the IBM Corporation.

Circle **no. 252** on reader service card.

dBASE II to C

the dBx translator

- **dBx** produces quality **C** direct from dBASE II or III programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- With power guidebook of conversion hints.
- Includes full screen handler and uses your current **C** database manager.
- May be used to move existing programs or help dBASE programmers learn **C** easily.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA. (Uses ANSI.SYS driver on MSDOS, CURSES under UNIX)
- Priced from $350, also available from distributors.

Desktop A.I.
1720 Post Rd. E. #3
Westport, CT 06880
(203) 255-3400

Circle **no. 258** on reader service card.

IQ C LISP

THE CLOSEST THING TO *COMMON LISP* AVAILABLE FOR YOUR PC

RICH SET OF DATA TYPES
Bignums, for high precision arithmetic
8087 support, for fast floating point
Arrays, for multidimensional data
Streams, for device-independent i/o
Packages, for partitioning large systems
Characters, strings, bit-arrays

FULL SET OF CONTROL PRIMITIVES
flet, labels, macrolet, for local functions
if, when, unless, case, cond, for conditionals
Keyword parameters, for flexibility
Multiple-valued functions, for clarity
Flavors, for object-oriented programming
Stacks, for coroutining
Closures, for encapsulation

LARGE COMPLEMENT OF FUNCTIONS
Mappers, for functional programming
format, for output control
sort, for user-specified predicates
Transcendental floating point functions
String handling functions
Over *400* functions altogether

APPLICATION SUPPORT
Save and restore full environments
User-specified initializations
Assembly language interface

HARDWARE REQUIREMENTS
8088 or 8086 CPU, MSDOS Operating System
390K RAM or more

IQCLISP
5¼'' diskettes and manual $300.00
Foreign orders add $30.00 for airmail.
U.S. Shipping included for prepaid orders.

∫q *Integral Quality*
P.O. Box 31970
Seattle, Washington 98103
(206) 527-2918
Washington State residents add sales tax.
VISA and MASTERCARD accepted.

EXTENDABILITY
defstruct, to add data types
Macros, to add control constructs
Read macros, to extend the input syntax
Extendable arithmetic system
Customizable window system

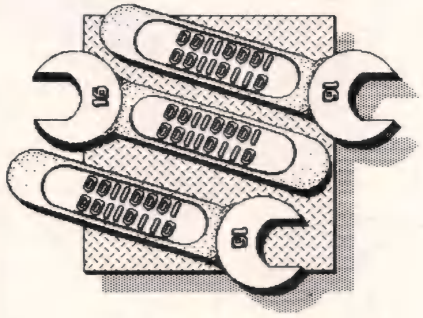
DEBUGGING SUPPORT
step, for single-stepping
trace, for monitoring
break, for probing
inspect, for exploring
Flexible error recovery
Customizable pretty-printer

MSDOS INTERFACE
Random access files
Hierarchical directory access
MSDOS calls

DOCUMENTATION
On-line documentation of functions
apropos
300-page indexed reference manual

16-BIT SOFTWARE TOOLBOX

by Ray Duncan



The DUP and FORCDUP Functions

MS-DOS functions *45H* (*DUP*) and *46H* (*FORCDUP*) have always been considered a little mysterious, except perhaps by those programmers who were nurtured under Unix. Both functions were added to MS-DOS in Version 2.0 at the same time as were the "extended file management" functions, and their documentation is a bit spare. The description in the *PC-DOS Technical Reference Manual* for function *45H* simply says that it "returns a new file handle that refers to the same file at the same position," and the explanation for function *46H* is that it "forces the handle in CX to refer to the same file at the same position as the handle in BX." In actuality, both of these functions are much more useful than the documentation suggests.

The *DUP* function (*45H*) is particularly convenient in applications that perform extensive file manipulation. Normally, the directory entry for a file is updated to reflect only the time and date last modified and the new length (if the file has been extended) when the file is closed. If your application extends a file and then crashes before closing the file, the new information at the end of the file is left floating in the form of lost clusters. Therefore, in programs that run for long periods, it would seem most wise to close and reopen a file whenever its length has been changed.

Unfortunately, the overhead of an open operation in MS-DOS is considerable, especially if the desired file is at the end of a fairly long path and is not in the current directory. You can avoid the open function altogether and still get your desired updating of the directory by *DUP*ing the handle for the open file and closing the duplicate. The close function turns out to be relatively fast in MS-DOS. See Listing One, page 106, for an example of this technique.

The *DUP*ed handle does subtract one from the maximum of 20 simultaneously active handles allowed for your process while it is open, but it doesn't count against the total number of open handles allowed for the system as a whole (the system total is set with the *files=* command in the config.sys file and defaults to eight).

The *FORCDUP* function (*46H*) can be used to redirect the input/output for any handle, previously opened to any logical device or file, to any other open device or file. The ramifications of this seem endless, but I suspect *FORCDUP*'s most common use is with the *EXEC* function to affect the behavior of the standard devices for child processes. Because the *open* handles of the parent program are inherited by the child, any desired redirection of the child's input or output can simply be put into effect at the parent's level before *EXEC* is called.

Jerry Jankura has been kind enough to donate a program that illustrates the use of *FORCDUP* to perform I/O redirection. It accompanies this month's column as Listing Two, page 106.

DOS Two-Point-What?

Last October, Microsoft released a revision of MS-DOS that hardly anyone has heard of—Version 2.25.

The main reason for MS-DOS 2.25's existence seems to be its enhanced character set support and interim character support, designed for the Far East OEMs that must support languages such as kanji and Korean. The *ASSIGN* and *LABEL* commands were added from MS-DOS, Version 3. In addition, MS-DOS 2's *DEBUG*, *SORT*, and *EDLIN* commands were replaced by MS-DOS 3.x's versions of the same. Many bugs reported in previous versions of MS-DOS (2.11 and earlier) were fixed.

Don't look for this version at your corner software store any time soon, though. Most U.S. OEMs appear to be ignoring it, even though it has less bugs, remains memory economical, and adds some of the desirable features of MS-DOS 3.x.

Windows Development Kit

Since its release late last year, Microsoft Windows has had surprisingly good market acceptance and in fact has been on the Softsel best-seller list for the last month as I write this. Although Windows is rather slow on the original 8088-based PC and is nearly unbearable without a hard disk, Windows on a PC/AT with an EGA is responsive and a pleasure to use. Prices for 80286-based PCs and fixed disks are decreasing rapidly, so it appears that if Windows was before its time hardwarewise, it was only just a little—though any significant penetration into the older PC user base will probably require the widespread availability of cheap turbo expansion boards and expanded memory boards.

Because of the dismal fates of ViSiOn and the PC version of GEM (seen any of those full-page color ads for GEM lately?), I was uncertain whether it was worth the time to pay any attention to Windows, how it works, and the machinations needed to write well-behaved Windows applications. The preliminary Windows development kit I received a year or so ago was intimidating to say the least, written as it was in the now-famous, infinitely self-referencing style of *Inside Macintosh*. To try and get some feeling for the future of

Windows, I attended Microsoft's Windows Developer Seminar in February in Seattle. I came away from this seminar with a changed outlook on Windows and what it portends for the future.

First, there is some confusion in the world of programmers about exactly what Windows is. Windows is not a desktop metaphor user interface like the one on the Mac. Icons are used in Windows only to symbolize tasks that are currently active in memory but do not have an open window or occasionally to select a resource (such as changing from one default disk drive to another). Icons are not used in Windows to represent and manipulate objects (files or programs) on a disk—you can't erase a file by dropping an icon in a wastebasket or copy a file by dragging an icon from one place to another, for example.

Windows is a multitasking executive, running on top of (and closely intertwined with) MS-DOS, that offers sophisticated memory management, dynamic loading and linking of code segments, intertask communication, a standardized virtual keyboard and pointing-device interface, and device-independent graphics services. Although Windows does have pull-down menus, tiled windows, scroll bars, and dialog boxes, these are in a way tangential to the intent and function of Windows. A pointing device can be used to advantage in Windows, but unlike the Mac, you can also get along quite nicely without one. The fact that well-behaved Windows applications will have a rather uniform user interface that dramatically shortens the learning curve for new users (as do Mac applications) can be viewed as just a nifty fringe benefit.

You are probably saying to yourself, "That all sounds great, but why should I as a programmer who uses MS-DOS worry about Windows now? Why not wait a year or two and see if Windows has any significant penetration of the user base I am concerned with and then decide whether to learn about its innards." You may be right. On the other hand, Microsoft made it clear at the seminar that much of the functionality of today's Windows (especially the multitasking and memory management) will be migrated downward into the MS-DOS kernel in future versions. In a sense, Windows can be thought of as a sneak preview of DOS 4 and 5 (in fact, the combination of MS-DOS 2 or 3 and Windows provides everything we were hoping for in the expected multitasking MS-DOS for the 8086/88-based PCs, and then some).

Windows apparently has even more significance for 80826-based PCs. Many of us have been a bit apprehensive about the upcoming Protected Mode versions of MS-DOS. Microsoft has been quite guarded on this topic until now, and the outlook has been further confused by leaks from IBM that it is developing its own Protected Mode operating system and by IBM's recent announcement that it is planning to use Digital Research's Concurrent DOS on a PC/AT-based point-of-sale product. At the seminar, Microsoft officials (including Steve Ballmer and Bill Gates) were suddenly surprisingly forthcoming with details about a Protected Mode MS-DOS. This may indicate that the major problems connected with this product have finally been solved.

During a panel discussion with members of the Windows development team and some outside Windows application developers, Gates asserted that the Protected Mode version of MS-DOS will be completely upward compatible with current MS-DOS versions and applications. Programs that are not well behaved (such as those that write directly to the video refresh buffer) will simply be executed in Real Mode and the fact that the operating system runs in Protected Mode will be invisible to them. In a way, this commitment to upward compatibility is somewhat unfortunate. Programs running in Real Mode, even under the control of a Protected Mode OS, can circumvent the 80286's mechanisms for protecting one task from another.

In other seminar sessions, guidelines were given for writing well-behaved programs under current versions of MS-DOS that will be able to run in Protected Mode on future versions and take full advantage of the 16-megabyte memory space. Ballmer, who has taken much of the flack for the many delays in Windows and was the author of the famous "before the snow falls" announcement, made a startling assertion. He said that well-behaved Windows applications created with the Windows development kit will run in Protected Mode on the upcoming PM version of MS-DOS without recompilation.

As for its own commitment to Windows, Microsoft laid it on the line in unmistakable terms. The company said that all future Microsoft applications (not languages) for the IBM PC that are not just evolutionary upgrades of existing packages will be Windows-dependent. Apparently, a port of Excel from the Mac to the PC is already underway for Windows. At first glance, such a policy seems a bit rash, but it may not be as risky as it sounds. The current Microsoft application packages for the PC (Word, Multiplan, and so forth) have been quite popular, and their quality is high; if future packages live up to the same standards, they may prove in themselves to be a potent driving force for Windows.

Those of us who attended the seminar each received a copy of the new retail release of the Windows Software Development Kit. This is a formidable package indeed, consisting of some 900 pages of typeset documentation in two volumes and a fistful of diskettes. The 12 floppies hold a special version of Windows with debugging support, Windows function libraries for C and Pascal, a library of macros for the folk determined to stick with assembler, an update to certain parts of the Microsoft C 3.0 compiler, a special linker, a modified SYMDEB that can be used with an external terminal or hooked to the PC's serial port, a dialog box editor, and so forth. A diskful of C source code for sample Windows applications is also included. Of course, in order to develop Windows programs, you must also buy the Microsoft C compiler, Pascal Compiler, or Macro Assembler separately.

The manuals for the development kit are nicely laid out and typeset but consist largely of reference material that is extremely dense. Only about a quarter of the material gives any guidance on the overall programming of a Windows application, and even in that section, it's rather difficult to see the forest for the trees. At the seminar, very helpful talks giving a more cosmic view were given by Microsoft programmers who have

Learn and Use AI Technology In Your First Evening With PROLOG-86

NOW ONLY $95

A complete ***Prolog Interpreter, Tutorial,*** and set of ***Sample Programs:***

☐ **Modify and write Expert Systems.**
Use the simple "Guess the animal" example on the Tutorial or use the sophisticated system for Section 318 of the US Tax Code written by one of the PROLOG-86 authors and published in the March, 1985 issue of Dr. Dobb's Journal.

☐ **Write Symbolic Math or Abstract Problem Solving Applications**
This is a complete Prolog program to convert from Farenheit to Centigrade: f_to_c(C,F):- C is(F-32) *5/9. Planning programs and games are included to help you learn.

☐ **Understand Natural Language**
Use the sample program that produces a dBase DISPLAY command as output.

☐ **BECOME FAMILIAR WITH PROLOG IN ONE EVENING.**

Programming experience is not required, but a logical mind is.

Prolog-86 Plus for $250 adds: Windowing, 8087, 640K memory access, random access files, strings support and definate clause grammar.

RECENT IMPROVEMENTS: Floating point support, MSDOS commands, on-line help, load Editor.

AVAILABILITY: All MSDOS, PCDOS systems.

Full refund if not satisfied during first 30 days.

ONLY $95

Solution Systems™

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle **no. 153** on reader service card.

LEARN LISP

Interactively and Write Real Programs with TransLISP for Only $75

A "COMMON LISP" compatible Tutorial, Interpreter, Debugging, and Pretty Printer plus a Fast, Full Screen Editor, Samples and Help

☐ **Start Easily and Quickly:**
A complete, modular tutorial helps you learn LISP at your own pace. An integrated, interactive environment provides all of the elements needed to enter, modify, analyze and debug programs.

☐ **Natural Language, Expert Systems and Symbolic Manipulation:**
Natural Language concepts are illustrated by a phone number retrieval program. Choose the best word processing program for you with the Expert System. Arithmetic expressions are translated to Assembler using a code generation program.

☐ **Write Realistic Programs:**
Short examples and substantial programs of about 10 pages in length help you learn by modifying, studying and using the key concepts needed to write programs of 1000 lines or more.

☐ **The "COMMON LISP" Standard:**
TransLISP includes a 300+ function subset of the "COMMON LISP" Standard. Use extras like the MSDOS interface and graphics. Or use "strict compatibility" to make programs written in TransLISP, upwardly compatible to work with other COMMON LISP systems like VAX LISP, GC/LISP or LISP Machine LISP.

Recent Improvements: 640K Memory use supports 12000 line programs. Full 8087 and 8086 floating point included.

Runs on any MS/PCDOS System with 256K. It is not copy protected.

"The best LISP I've ever used. Three times before, I tried to learn LISP, but until now I could never break through the surface." *W.L. Whipple, User*

Full refund if not satisfied during first 30 days.

ONLY $75

Solution Systems™

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle **no. 155** on reader service card.

16-BIT

(continued from page 113)

been working within the Windows environment for two years or more and an additional volume of proceedings, programming guidelines, debugging examples, and sample source code was given to each attendee. Taken together, these remove much of the start-up fright factor from coding for Windows. If you are considering or embarking on a Windows-specific application, I'd highly recommend that you sign up for one of the scheduled developer's seminars—it will save you dollars and hours in the long run.

At this time, the ante for programmers who want to work with Windows is high. By the time you add up the cost of a PC/AT with 512K RAM, a hard disk and an EGA (the development configuration I would recommend), the Windows development kit, the Microsoft C compiler, and possibly a trip to Bellevue or Boston to attend the Microsoft Windows classes, you are talking about a lot of money. And if you aren't a C programmer, you're largely out of luck for the present. The Pascal and assembler support for Windows development seems rather half-hearted at best, and there are no bindings at all for FORTRAN, COBOL, or BASIC compilers.

If Microsoft is really committed to get Windows moving among software developers, I have a few suggestions for things it could do relatively quickly: release a Windows-specific version of BASIC similar to Mac BASIC; release a low-cost set of QuickBASIC bindings to Windows; release a lower cost, simplified Windows development kit for C programmers; and release a Turbo Pascal Windows toolkit. All these things should be priced around $100 to remove them from the "I wonder if I can talk my company into buying this" category and put them in the MasterCard/Visa, impulse-buy category. Of course, I am writing this in February, so when you read this in June, some lower-cost developer products may already be history.

The development kit can be ordered directly from the Microsoft Telemarketing Group [(800) 426-9400] and currently costs $500.

Building Overlays

Dr. Glenn Roberts of the Mitre Corp. responded to David Rabber's request for information on the overlay capability of the Microsoft linker (December 1985 column). He writes: "We obtained Version 3.01 of the Microsoft linker as part of the Microsoft C compiler package. This version of the linker supports overlays and the following information on it is condensed from the Microsoft documentation.

"You specify overlays in the list of modules that you submit to the linker by enclosing them in parentheses. Each parenthetical list represents one overlay. As an example, if the following were your response to the 'Object Modules' prompt:

Object Modules [.OBJ]:
a+(b+c)+(e+f)+g+(i)

then (*b*+*c*), (*e*+*f*), and *i* are overlays.

"Some pertinent notes:

- Overlays are loaded into the same region of memory, so only one can be resident at a time.
- Duplicate names in different overlays are not supported, so each module can occur only once in a program.
- The linker replaces calls from the root to an overlay and calls from an overlay to another overlay with a software interrupt, followed by the module identifier and offset. The default interrupt for calling this overlay manager is *03FH*.
- The names of the overlays are appended to the EXE file, and the name of this file is encoded into the program so that the overlay manager can access it. If the manager cannot find this file, it will prompt you for the file's name. After you've supplied the name, you can later swap disks in the associated drive. The overlay manager will detect this when it needs an overlay that is on a disk that has been removed and will prompt you to replace the disk and 'strike any key when ready.'
- The overlay manager is smart enough to search the current path for the EXE file.
- Control to overlay modules must be passed through far call/return sequences because the linker finds these and replaces them with the overlay interrupt. This rules out the use of indirect calls across overlays via pointers.
- You can change the default interrupt used to call the overlay manager using a switch on the linker:

/OVERLAYINTERRUPT:number

where *number* can be 0–0FFH.

"I should mention that I haven't experimented with the overlaying capabilities of this linker. I've merely stated, in condensed form, the information in the Microsoft documentation."

Another Resource for Programmers

The Programmer's Journal, edited by Robert Keller, is rapidly developing into a sort of modern-day equivalent of the original *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia*. Casual and gossipy, yet stuffed with useful information, it definitely deserves a look. Contact the magazine at P.O. Box 30160, Eugene, OR 97403; (503) 484-2162.

DDJ on CompuServe

One of the Data Libraries (DL2) on the CompuServe DDJ Forum is devoted to the 16-Bit Software Toolbox, and most of the program listings published here in the last year or so are already available for downloading. If there are particular programs from farther back in the history of this column that you would like to see placed on the DL, please let me know. Also, I'd like to encourage everyone to use the DDJ Forum to send me comments, suggestions, criticisms, and programs. I guarantee quick response!

DDJ

(Listings begin on page 106.)

Vote for your favorite feature/article.
Circle Reader Service **No. 6.**

Instant-C™: The *Fastest* Interpreter for C

Runs your programs 50 to 500 times faster than any other C language interpreter.

Any C interpreter can save you compile and link time when developing your programs. But only ***Instant-C*** saves your time by running your program at compiled-code speed.

Fastest Development. A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with ***Instant-C***. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. *Only* ***Instant-C*** *will let you edit, test, and debug at the fastest possible speeds.*

Fastest Testing. ***Instant-C*** immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

Fastest Debugging. ***Instant-C*** gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional breakpoints throughout your program. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

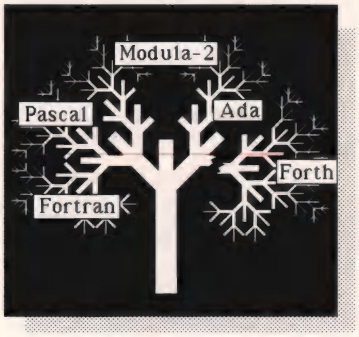
Fastest Programming. ***Instant-C*** can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86. ***Instant-C*** *gives you working, well-tested programs faster than any other programming tool.* Satisfaction guaranteed, or your money back in first 31 days. ***Instant-C*** is $495.

Rational
Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

Circle **no. 145** on reader service card.

STRUCTURED PROGRAMMING

Namir Clement Shammas



In this column dedicated to Pascal, Ada, and Modula-2—descendants of the ALGOL language—I will discuss language and implementation issues as well as applications written in them. The livelihood of any column draws from readers' interaction. *DDJ*'s CompuServe forum is an excellent place for fast feedback and dialogue; the U.S. mail is the slower alternative. You are invited to share your tips, tricks, and programming techniques.

In this issue I'll discuss two topics. The first part of the column deals with simulating overloaded procedures and functions in Pascal and Modula-2. The second part looks at exporting opaque types and data hiding in Modula-2.

Overloading Procedures

An overloaded procedure or function is one that exists simultaneously in several different versions within the same program. This allows you to use the same procedure call with several different kinds of arguments. An example of this is the Pascal intrinsic *Writeline( )*, which can take any number of arguments with many different data types. Unfortunately, Pascal and Modula-2 do not allow you to write overloaded routines explicitly because you're not allowed to create two functions or procedures with the same name in the same code body.

A Modula-2 program can import different libraries that may contain procedures with the same name. Thus you can import an entire library and use the overloaded procedure prefixed with the library name. Consider, for example, two library modules *RealInOut* and (a fictitious) *LongRealInOut* that take types *REAL* and *LONGREAL*, respectively, and both contain a procedure called *WriteReal( )*. To use the overloaded *WriteReal( )* procedure, you can call *RealInOut.WriteReal( )* or *LongRealInOut.WriteReal( )*. Because the two procedures called *WriteReal* are in different modules, the Modula-2 compiler is able to accept them.

Variant records provide a way to create "simulated" overloaded procedures. The simulation stems from the fact that there is really only one copy of the procedure. Admittedly, a bit more effort is required to make such procedures readable. The variant parts of a record enable the program to tackle different data items varying in basic type or number. I'll use three examples. The first overloads a routine that handles arrays of different basic data types. In this case, the macro structures are similar or identical but the micro structures are different. The second case deals with representing the same information with alternate notations. The third case shows a stack containing data structures of several types.

The first example (see Listing One, page 108) shows a Pascal procedure to perform a histogram count. In general, the input is an array of data items accompanied by an array of perfectly sorted histogram bin limits. Each bin limit gives the upper and lower bounds for the values to fall within one of the output slots of the histogram. This gives the flexibility that the histogram bin sizes need not be equal. Data values lying outside the histogram limits are ignored. The procedure *Count_Histogram* is capable of handling arrays of *REAL* as well as arrays of *STRING* data types. For the latter type, the variant records supply additional information. They include two integers that mark the first and last characters (within each string) of the substrings to be used for the bin comparison. Notice that the *Count* array is the only output in the variant record. Procedure *Count_Histogram* has its own local procedures to perform the frequency count for each different data type. You can easily add similar procedures to handle arrays of integers or characters.

Similar routines can be written to implement various searching and sorting techniques. In a future column I will discuss generic sorting. Generic routines provide a flexible solution to handle a wider variation in data types.

The second example, shown in Listing Two, page 108, looks at the situation in which information can be represented by alternate notations. You can represent a complex number (that is, a point on a two-dimensional graph) either by using rectangular coordinates (x and y) or by polar coordinates (modulus and angle). Thus you can have two sets of data each consisting of two *REAL* numbers. To process the information you must know what sort of coordinates are supplied. Listing Two shows a simple Pascal procedure to add two complex numbers. Each of the numbers can be supplied to the procedure as rectangular or polar coordinates, indicated by the *Is_Polar* field. Similarly the output can be obtained in either coordinate system. The example can be extended to systems of three or more dimensions.

The third example presents a stack that handles a variety of data types. Here, the differently typed items are more logically related. Compared with the histogram count example in which different data types are handled in parallel, this one handles them in series.

The fields of the variant portion contain the same number of identifi-

ers; only the types are different. Listing Three, page 108, shows three Pascal routines to push, pop, and selectively pop stack items. The nature of stack and queue manipulations permits them to accept multitype data in certain applications. Notice that the variant record contains user-defined record structures. You can add more variant fields without changing the code for the procedures. Unordered lists (single- and double-linked) can be constructed in a similar manner.

Exporting Opaque Types and Data Hiding

An opaque data type is one that includes no representation of the internal structure of the data. An example is the type *REAL*, the internal structure of which (the exponent and mantissa, along with their signs) is not available to the programmer. The Modula-2 feature of exporting opaque types and data hiding (sometimes referred to in Modula-2 books as data abstraction) has been with us all along, but originally it was a luxury only compiler writers enjoyed. It was impossible for us to use a data type exported from another module or library without explicitly stating its internal structure. Now this situation has changed with developments in software and hardware, and Modula-2 offers similar privileges to library module developers. This is done by having the definition module state the exported data type name only, with no structure definition. Hence, the opaque type is born. The implementation module has the complete type definition along with all the routines to manipulate it. Modula-2 requires that opaque types be defined as pointers to other data types. The client programs importing the opaque types do not have access to their internal structure, and thus they cannot have their own procedures to manipulate the opaque types. The library developer is responsible for providing every routine needed!

By hiding the internal structure of an opaque type, library authors can modify it, and the procedure bodies, without affecting client programs. They may want to do this for a variety of reasons, such as prototyping or discovering a superior or more convenient alternate structure.

Applications for exporting opaque types are numerous. The simplest example is string libraries. Table 1, below, shows four alternative definitions for an opaque string type. The first three string types use a finite array to store characters. The fourth type uses true dynamic dimensioning by employing the imported type *ADDRESS*.

The type *string1* is straightforward. The implementation procedures must rely on ASCII zero code as the string terminator for partially filled strings. The type *string2* incorporates a string length counter. Using it along with the predefined *HIGH( )* function, which returns the upper bound of the character array, the appropriate string lengths are managed. Exporting this type as transparent may cause problems with user-written procedures that corrupt the length counter, and thus the use of an opaque type in this situation is more attractive and justifiable. The third type is a slight modification of *string2*, adding a total string counter. Ford and Wiener[1] discuss this string type and point out that the structure uses the total length field dynamical-

```
CONST MaxLength = 255; (* or any other length, up to 65535 *)
(* Alternative # 1 *)
TYPE string1 = POINTER TO RECORD
                    strch : ARRAY [0..MaxLength] OF CHAR
               END;
(* Alternative # 2 *)
TYPE string2 = POINTER TO RECORD
                    long : CARDINAL;
                    strch : ARRAY [0..MaxLength] OF CHAR
               END;
(* Alternative # 3 *)
TYPE string3 = POINTER TO RECORD
                    long,
                    TotalLength : CARDINAL;
                    strch : ARRAY [0..MaxLength] OF CHAR
               END;
(* Alternative # 4 *)
(* Note: ADDRESS type is imported from module SYSTEM *)
TYPE string4 = POINTER TO RECORD
                    long,
                    TotalLength : CARDINAL;
                    strch : ADDRESS
               END;
```

Table 1: *Alternative Modula-2 opaque string structures*

```
(* Matrix may have negative indices *)
TYPE Matrix1 = POINTER TO RECORD
                    FirstRowIndex,
                    LastRowIndex,
                    FirstColumIndex
                    LastColumIndex : INTEGER;
                    (* ADDRESS is Imported from SYSTEM *)
                    MatrixMember : ADDRESS
               END;
(* Matrix with zero or positive indices *)
TYPE Matrix2 = POINTER TO RECORD
                    LastRowIndex ,
                    LastColumIndex : CARDINAL;
                    (* ADDRESS is Imported from SYSTEM *)
                    MatrixMember : ADDRESS
               END;
```

Table 2: *Dynamic opaque matrix structure*

(continued from page 117)

ly. The *ALLOCATE( )* procedure is used instead of *NEW( )* to accomplish the above task. Employing *ALLOCATE* forces the run-time system to create a dynamic structure according to the actual record size, which may be smaller than the maximum allowable size. The fourth string type differs from the others in that none of its fields is an array of characters. Like *string3* it contains fields to keep track of the current string length and the total size is dynamically allocated. The field of type *ADDRESS* is the pointer that locates the actual character string. The advantage of type *string4* is the creation of strings with tailored sizes.

Another example of alternative representation is complex numbers, discussed earlier. It is possible to have two library implementation modules: one for rectangular coordinates, the other for polar coordinates (see Ford and Weiner: 177). Because opaque types are involved, procedures to create and return the real and imaginary parts of a complex number must be supplied to client programs.

Modula-2 supports only one-dimensional open arrays in procedure arguments. Ford and Wiener present a dynamic matrix library exported as an opaque data type. The matrix is defined as a pointer to a record that contains the upper and lower dimension limits and an identifier of type *ADDRESS*, as shown in Table 2, page 117. This structure allows you to create matrices tailored to size, although speed is on the slow side.

Other popular data structures such as binary and B-trees can also be exported as opaque types. Multiway trees such as the B-tree, B+ Tree, B* Tree,[2] and B++ Tree[3] examples of complex data structures. Library database developers may start by exporting a B-tree structure as an opaque type. Hiding the exact structure gives them the ability to select one of the above structures or implement their own refinements, which may involve adding more pointers or resizing the B-tree page. One problem generally encountered with such data types occurs when you perform I/O with files. As the data structure changes, the new library version must be able to identify and read previous structures saved in files.

The simple example presented in Listing Four, page 109, deals with a module exporting procedures to simulate a basic RPN calculator with four stack registers (*X*, *Y*, *Z*, and *T*) and a *LASTX* register, similar to a Hewlett-Packard calculator. Parts A, B, and C of the listing show the definition module and the two implementation modules, respectively. In part B the stack is formed by five scalar identifiers, whereas part C shows an array representation. The zeroth member corresponds to the *LASTX* register, the first to the *X* register, and so on. It is interesting to note that, while using the array representation, a *FOR* loop can be used to push and pop the stack. The scalar representation is more readable, however. The *HPStackMod* module exports four basic operations—number entry, stack clearing, recalling *LASTX* into the *X* register, and a function to return the *X* register. The latter function, which may seem extremely trivial, is nevertheless essential because of the use of an opaque type to represent the stack.

An Invitation

I encourage you to send me short utility routines or programs that perform useful tasks—for example, tapping into hardware and operating systems. I'm also looking forward to the validation of the IBM PC/AT Ada compiler by Alsys Inc. Obtaining a copy of this will help my discussions about the language.

Notes

1. G. Ford and R. Wiener, *Modula-2: A Software Development Approach* (New York: John Wiley & Sons, 1985).
2. M. Loomis, *Data Management and File Processing* (Englewood Cliffs, N.J.: Prentice-Hall, 1983).
3. N. Shammas, "B+ trees, B++ trees, and statistics in AI," *Computer Language*, 2 (6) (1985): 13–18. **DDJ**

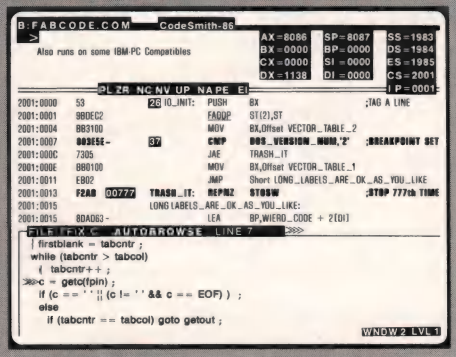
(Listings begin on page 108.)

Vote for your favorite feature/article.
Circle Reader Service **No. 7.**

IBM PC/XT/AT SYMBOLIC DEBUGGER

"C" FOR YOURSELF!

ONLY CodeSmith™-86 has AutoBrowse®



The *effortless* way to debug your C, PASCAL, & FORTRAN.

Step & Break thru synced SOURCE and Machine CODE WINDOWS.

☑ Dual-mode PATCHING ASSEMBLER • 256K RAM min.
☑ 4 STOP-on-Data-Compare conditions • *DOS 2.0 → 3.x*

CodeSmith-86 *the original* Multi-Window Debugger

GUARANTEED 30 DAYS OR YOUR BUCKS BACK

"CodeSmith really shines..." —*PC TECH JOURNAL*

"It has greatly speeded up the development of new programs and the maintenance of existing programs." —*MicroPro Int'l Corp.*

"...has saved us uncountable hours in debugging communications software." —*Dave Adkins, Jim Dukat, Los Altos CA*

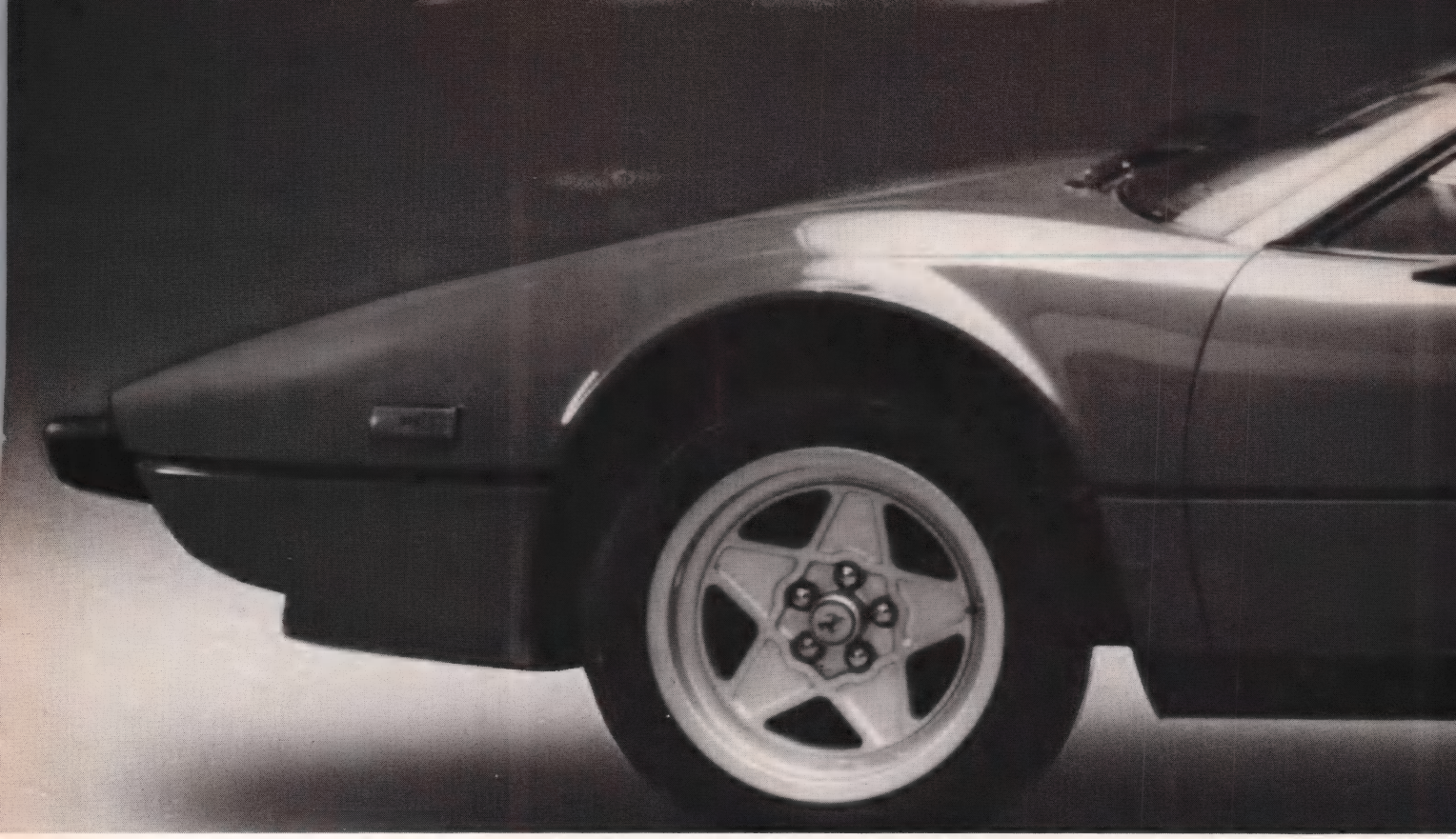
"...has almost all the features that are in the Hewlett-Packard 64000 emulator I use at work." —*Joel Lagerquist, Riverside CA*

"CodeSmith-86...a superb utility." —*Steven Kang, New York NY*

VISUAL AGE ◆◆ In Calif. (213) 534-4202
642 N. Larchmont Blvd. ★ Los Angeles, CA 90004

ORDER NOW (800) 732-2345
MasterCard VISA
$145.00

Circle **no. 291** on reader service card.



C Programmers! May PforCe Be With You.

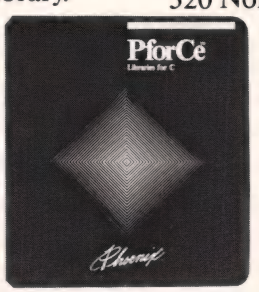
Writing in C? Half your job could be done already. With PforCe.™ The first library of object-oriented C functions and subsystems. Written in C. Fully integrated, optimized, debugged, and ready to go.

High level functions that let you manipulate objects like windows. Fields. Screens. Menus. Change an object's characteristics globally. Or tailor them to a specific application. So you can write code faster. And more economically.

Low level functions to give you complete hardware control. Defaults you can change at will. Plus, more sophisticated subsystems to handle complex tasks. A database system with demand paging and B-trees to store, access and index data. And, since PforCe includes source code, you can modify anything in the library.

But that's not all. PforCe is more than a program-generation toolbox. It's designed to make things easy to find and use. Alphabetically. By functional group. Or, while you're editing through a pop-up utility. And PforCe comes with an easy to follow tutorial to help you become more productive quickly.

PforCe is available for Microsoft,® Lattice,® CI86,™ and Wizard™ compilers. All memory models of each compiler are supported. You can use PforCe with any supported compiler by re-compiling the source code, but we provide a pre compiled version for the compiler that you specify at order time. $475 complete.



Special Introductory Offer.

You would have to spend $700–$800 on several libraries to get the range of functionality provided by PforCe. But we want you to be convinced.

Order before June 15, 1986 and save an additional $80 on the regular price of $475. **Special offer $395.**

Send for your PforCe information kit today. Call or write:

Phoenix Computer Products Corporation
320 Norwood Park South
Norwood, MA 02062
(800) 344-7200.
In Massachusetts (617) 762-5030

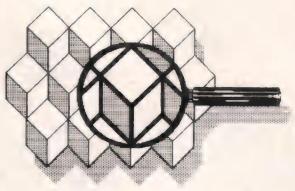
Programmers' Pfantasies™
by

Phoenix

Programmers' Pfantasies and PforCe is a trademark of Phoenix Computer Products Corporation. Microsoft is a registered trademark of Microsoft Corporation. Lattice is a registered trademark of Lattice, Inc. CI86 is a trademark of Computer Innovations. Wizard is a trademark of Wizard System Software.

Circle **no. 91** on reader service card.

OF INTEREST



Corporations today face the same problems that motivated the Department of Defense (DOD) to create a standard programming language and eliminate the proliferation of languages and dialects that contribute to astronomical software upkeep costs, estimated at 80 percent of total systems maintenance. Mandated for all DOD mission-critical applications, the Ada programming language is expected to have increasing importance for all commercial applications. Historically, the lack of compilers has inhibited the widespread use of Ada.

An Ada compiler for the IBM PC/AT from **Alsys** enables full-scale Ada application programs to be written for the PC. Through the use of protected (virtual) mode, the compiler permits an application program to overcome the 640K limitation imposed by the DOS operating system and to access extended memory (up to 16 megabytes on the PC/AT). The compiler is packaged with a 4-megabyte memory board. It also features 8086 or 80286 instruction, an online help facility, and error checking. The compiler for the IBM PC/AT is priced at $3,000.

Tartan Laboratories has developed a C programming language compiler for the IBM PC/RT. Both an Ada and a Modula-2 compiler for the IBM PC/RT are currently under development.

An Ada compiler system for the IBM PC, PC/XT, PC/AT, and compatibles is available for $895 from **Artek Corp.** The compiler system meets virtually all the latest DOD specifications except "tasking" and runs under the MS-DOS or PC-DOS operating system on PC-compatible computers having at least 384K of memory. Hard-disk mass storage is recommended for the development of large applications. Demonstration disks are offered for $29.95. The full system is available to buyers of the demonstration disk for $29.95 less than the regular price.

Artificial Intelligence

Microsoft's LISP 5.1 offers more primitives, greater capacity, expanded arithmetic, improved debugging, and faster list sorting than do earlier versions. It also features common LISP support and split-screen capabilities. Minimum system requirements for Version 5.1 are a PC running MS-DOS or PC-DOS 2.0 or later, 128K of memory (although Microsoft recommends at least 256K), and one disk drive (two are recommended). It has a suggested retail price of $250.

OPS83, the high-performance expert systems programming language from **Production Systems Technologies**, is available for use on the IBM PC and compatibles. This version of OPS83 is identical to the original version introduced in 1984 for use under VMS and Unix on the VAX series machines and Apollo Domain. Recently, it has been made available for use on the MicroVAX, Sun Workstation, and AT&T's 3B series. It retails for $1,950.

An integrated systems development environment for planning, analyzing, designing, and constructing computer-based information systems is available from **KnowledgeWare.** Called the Information Engineering Workbench, this software family uses expert system and computer-aided design and programming techniques to automate information engineering. The new family includes an integrated set of diagramming tools for several common diagram types, including entity, decomposition, data-flow, and action diagrams.

C Language

Raima Corp. has announced Version 2.1 of db_Vista, its database management system for software development in the C programming language. It is designed for use with MS-DOS or Unix-like operating systems. The new version features improved B-tree key field handling; a key-file rebuild utililty; a database consistency check utility; a data-field alignment check utililty; and file-transfer utilities for dBASE, R:base, and ASCII files. The db_Vista multiuser version costs $990 with source and $495 without source. The single-user version is available for $495 with source and $195 without.

High C, a C cross compiler implemented for VAX/VMS running on the Intel 8086/88/186/188/286 family of microprocessors, is available from **Microtec Research.** High C features support for ROMable code for embedded applications, nested functions complete with up-level references, nested functions passable as parameters, a full set of memory models, three integer ranges, and three IEEE real precisions. The product also contains many compiler controls and options, including one for strict ANSI standard checking. The complete High C software package costs $7,000 and operates on DEC VAX under VMS.

Computer Innovations has released a free booklet on its Optimizing C86 C compiler. The features of the C86 discussed include language conformance, Unix compatibility, and source-level debugging support. The booklet also features a complete listing of run-time options and functions.

Fast Programming from **Subject, Wills & Co.** is a C generator tool for business-application developers. The product includes a B+ tree index facility, a field-independent record management system, a complete set of run-time utilities, a library of C routines, and several C program generators. Fast Programming sells for $995 for a site license. It is available for PC-DOS and Xenix on the IBM PC/AT and Unix V on the AT&T 3B2/3B5 computer line.

Version 2.0 of Bastoc from **JMI Software Consultants** translates BASIC programs into C. Bastoc analyzes the use of numeric variables to determine which floating-point variables can be replaced by integer variables. Additional optimizations include eliminating unreachable code; converting BASIC assignment statements, where possible, into simpler increment or decrement operations available in C; and evaluating string expressions at compile

time. The product includes a BASIC compiler program. Binary versions are available for the IBM PC and compatible systems using MS-DOS, the AT&T 3B2 (Unix V), the AT&T Unix PC/3B1 (Unix V), the Radio Shack Model 16 (Xenix), Sperry 5000 (Unix V), and several additional Unix and Unix-like systems. The price for single-user systems is $495.

Application Development

Version 6 of **Netron's** computer-automated programming development software for the Wang VS includes a feature that allows automatic control of in-house screen design standards for file-maintenance programs. The new version also adds background processing from user-defined function keys and supports use of qualified data names. The program includes a built-in standard ANSI 74 COBOL and is suitable for running production systems of any size and complexity.

The **Oasys** 68020 Toolkit features a complete line of compilers (C, Pascal, FORTRAN-77), assemblers (including linker, loader and librarian), debuggers, simulators, profilers, real-time OS, and down-line load utilities. Support for the 68881 floating-point processor is also provided. The Toolkit is available for DEC VAX (VMS, Ultrix, Unix), DEC MicroVAX (VMS, Ultrix), Sun, Apollo, Pyramid, PCs, and other 68000 and 32000 systems running Unix. A typical configuration of a C compiler, assembler, linker, and librarian starts at $3,200 in single quantities.

Release 5.0 of **STSC's** APL*Plus PC System adds speed to the development process with its APL language notation. The run-time version is an adaptation of the APL*Plus PC system specially modified to run a single application. This modified interpreter enables developers to include enough of the APL*Plus system to run their applications but not enough to allow end-users to write or modify their own APL programs. The run-time system is licensed on a royalty or per-copy basis.

Release One, Version 3.06, of Q'Nial from **Nial Systems** is a high-level interactive interpreter that handles symbolic and numeric computation with equal facility. It is used primarily in logic programming and other artificial intelligence applications. A Q'Nial license costs $300 for PC/XT/AT versions. The entire package, including media and shipping, costs $375. Educational licenses are half-price. A site license for educational institutions costs $500.

For the IBM PC

DSD86 from **Soft Advances** is a full-screen symbolic debugging program for IBM PC-compatible computers running PC-DOS or MS-DOS. DSD86 offers a built-in windowing system for a user-controlled screen layout with six different display types, including instructions, registers, stack, memory, and source. The keyboard interface can be customized, permitting arbitrary command lines to be bound to any Ctrl, Alt, or function key. A recursive

C Cross Compiler 68000/08/10/20

Now Generates: Position Independent Code

Features:

- Full, Standard C
- Easy to Use Compiler Options
- Complete User Documentation
- Global Code Optimization
- Optional Register Allocation Via Coloring
- ROMable and Reentrant Code
- Comprehensive Royalty Free Run-time Library
- Floating Point Library Routines
- Intermix MCC68K C with ASM68K Assembly Language or Microtec PAS68K Pascal
- Optional Assembly Language Listing Intermixed with MCC68K C Source Line Number
- Symbolic Debug Capability

The Microtec MCC68K C Cross Compiler is a complete implementation of the 'C' programming language as defined in The C Programming Language by Kernighan and Ritchie with extensions.

MCC68K emits highly optimized assembly language code for the Microtec ASM68K Motorola compatible assembler.

The Microtec MCC68K package includes the compiler, relocatable macro assembler, linking loader, run-time library, and comprehensive user's guide.

Host computers include: DEC VAX, DG MV-Series, Apollo, IBM PC and PC-compatibles..

We're **Functional** and **Fast** and **Serious** about our products. We've been providing flexible and economical solutions for software developers since 1974.

Beginning with product concept, through development, quality assurance, and post-sales support - **Quality**, **Compatibility** and **Service** are the differences which set Microtec Research apart from others.

If you're a serious software developer, shopping for software development tools, call or write today for more information:

800-551-5554,
In CA call (408) 733-2919.

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
Mailing Address: P.O. Box 60337, Sunnyvale, CA 94088

MICROTEC® RESEARCH

(continued from page 121)

macro facility allows consistent extensions to the set of 45 commands provided by DSD86. The list price is $69.95.

Programmers for the IBM System/38 computer can write and edit RPGIII source code on an IBM-compatible personal computer using the Baby/38 Source Entry Utility (SEU), a software package from **California Software Products.** Baby/38 SEU emulates the System/38 source entry utility to provide full-screen editing without the expense of System/38 hardware. Moving editing functions off-line to a PC frees the System/38 for other tasks and permits programming to continue even if the system is down. Baby/38 SEU requires an IBM or fully compatible PC with a minimum of 384K memory, DOS 2.0 or later, parallel printer port, and dual-floppy or floppy- and hard-disk drives.

Applied Data Research has released Version 2.0 of ADR/PC Datacom, a PC-based query and report writing facility. PC Datacom supports the exchange of data between an IBM PC and a mainframe as well as other PC functions. The new version features PC-based query creation and host data download and upload, a full-function report writer, data export and import for the exchange of data between PC spreadsheets and other application software, and a procedure facility for unattended and repetitive tasks. The product operates on any standard IBM PC, PC/XT, or PC/AT computer using PC-DOS 2.0 or later. It requires a minimum of 512K memory and two dual-sided, floppy-disk drives or a hard-disk drive. The IBM 3270 PC is also supported and requires 640K of memory.

Flagstaff Engineering has announced three products—File Connection, Word Connection, and Tape Connection—for data/text transfer to and from the IBM PC or compatibles. File Connection is a 3½-, 5¼-, and 8-inch disk subsystem that interfaces to a PC and allows users to transfer files from many different systems. Word Connection allows transfers between different word-processing systems, such as Displaywriter, Lanier, OS/6, NBI, Wang, Xerox 860, CPT, Microsoft Word, Multimate, and WordStar. The Tape Connection is a half-inch magnetic tape drive interface that allows transfer of files from a PC to tape and back.

Maxit, a memory card with software that expands available memory on an IBM PC, PC/XT, PC/AT, or compatible computer, is available from **McGraw-Hill Software.** Maxit requires DOS 2.0 or later and can fill out the memory of the IBM PC/AT, taking it from 512K to 640K and beyond. Maxit is priced at $195.

Hallock Systems has announced three enhancements to its Pro68 product line. DOS68 is a PC-DOS-compatible operating system designed for use on the Pro68 or Pro68/10 coprocessor cards. It is available for use with C, Pascal, Forth, BASIC, and FORTRAN. The system sells for $150. Pro68/10 is a single printed-circuit card that can be installed in any full-size PC, PC/XT, or PC/AT bus slot. The card includes a 68010 microprocessor running at 12 MHz, up to 1,024K of on-board 16-bit parity-checked memory, provisions for a 6-MHz math processor, two serial I/O channels, a 16-bit 680x0 expansion bus, and a proprietary dual-ported PC bus interface. Pro68/10 is available in two configurations, costing from $1,995 for the 512K version and from $2,195 for the 1,024K version. RTX68 is a time-sliced multitasking executive that supports up to 256 concurrent tasks. Each task is assigned one of 256 possible priority levels that can be changed during run time on a dynamic basis. RTX68 is designed to run concurrently with the host system PC-DOS. It is available for $150.

The IBM PC-compatible RS-232 5¼-inch Floppy Data Storage and Transfer System is available from **Analog & Digital Peripherals.** It features host and/or manual controls, ASCII or full binary operation, baud rates switch selectable from 110 baud to 19.2K baud, and automatic data verification. It is available in 110 VAC stand-alone or OEM configurations. The stand-alone system is priced at $1,095.

Communications

Quadram has launched its MainLink line of micro-to-mainframe communications solutions with four 3278/79 emulation products. Two of the products, the MainLink Standard coaxial connection and the MainLink Plus coaxial connection, link directly to an IBM 3274 or 3276 cluster controller for local or remote processing. Both are Irma compatible. They are also equipped with soft-loaded microcode, permitting upgrades to be made with a floppy disk. The MainLink Standard remote and MainLink Plus remote attach via synchronous modem to an IBM 3705, 3725, or equivalent communications controller in SNA/SDLC mode. Both permit emulation of an IBM 3274 cluster controller and 3287 host-addressable printer. The four products retail as follows: MainLink Standard coaxial, $895; MainLink Plus coaxial, $1,145; MainLink Standard remote, $545; MainLink Plus remote, $985.

SoftCraft has a new release of its Btrieve file management software for the IBM PC/AT and compatibles. Btrieve 4.0 and Btrieve/N 4.0 (for multiuser and LAN systems) feature variable-length records, data encryption, password protection, and a file-level verify option. Both are for software development in BASIC, Pascal, COBOL, C, FORTRAN, Modula-2, and APL. They cost $245 and $595, respectively.

Network-OS 6.0, a Netbios-compatible, network operating system, supports DOS 3.1, all major network topologies, and Novell file and record locking. Available from **CBIS**, Network-OS is menu-driven, and commands are presented on hierarchical, pull-down screens. LAN resources are addressed by user-defined object names and mapped by mouse or keyboard. The retail list price of Network-OS is $995. Interface boards cost $295.

Lamar Micro has developed a 65C02 cross assembler program for the Atari 520 ST on Atari format disk. This program, called C02 Cross Assembler, allows the Atari to act as a software development system for Apple, Atari, and Commodore computers that use the 6502 or 65C02 microprocessor. The price of the program is $89.95.

TDI Software has released a Modula-2 for the Commodore Amiga. The software features full in-

terface to ROM Kernal, Intuition and AmigaDOS, 32-bit native code implementation, support for transcendental functions and real numbers, separate compilation of modules with version control, Code statement for inline assembly code, and the ability to locate and identify errors in source code. The Modula-2 comes in regular and developer's versions. The developer's version has an extra disk containing all the definition module sources, a symbol file decoder, link and load file disassemblers, a source file cross-referencer, the kermit file-transfer utility, and the source code for several of the Amiga modules. The retail price of the regular version is $89.95; the developer's version is $149.95.

Peachtree Technology has introduced the T-33e Back-Up Subsystem. The T-33e utilizes the existing external floppy port on any IBM PC, PC/XT, PC/AT, or compatible. It is MS-DOS-compatible and can back up 30 megabytes. An LED readout provides users with tracking, power, and drive information and offers self-diagnostic capabilities, including an on-board error-detection device. The T-33e retails for $795 and comes with two 10-megabyte reels. It also comes in an internal half-height configuration that retails for $695.

Mastercom-Telecommunications Utility is a smart-terminal and file-transfer utility available for the IBM PCjr and most IBM PC-DOS- and CP/M-80-compatible computers. Mastercom, available from **The Software Store**, is designed to capture data onto a disk and/or printer, send files, and transfer files using the Christensen XMODEM error-correcting protocol. It includes auto-dial, auto-answer, host-mode unattended operation, batch-file transfer, directory display, file erase, file rename, disk-drive logging, stored responses, and more.

Samsung Semiconductor has introduced two families of high-performance CMOS logic products: the 54/74 Advanced High-Speed CMOS and the 54/74 High-Speed CMOS. The two product families contain 63 devices, including octal buffers, octal transceivers, octal latches, and inverters. They also feature low power dissipation, high levels of noise immunity, low input currents, wide operating voltage supply and temperature ranges, 4,000V ESD protection, and the ability to handle latchup trigger currents above 200 ma.

The Epsilon Extension Language from **Lugaru Software** is an interpreted, dynamically linked extension language that resembles C, augmented with functions and variables to facilitate writing editor extensions. It features source code for all commands, unlimited file size, an on-line tutorial, an EMACS-style command set, language support, command-name and file-name completion, and full DOS path support. Epsilon's price is $195.

Watcom Products has released Maple, an interactive system for algebraic computation. Maple provides diagnostic and debugging facilities and supports two output formats: two-dimensional, multiline format and one-dimensional, line-printing mode. It is available for IBM VM/SP CMS, Digital VAX/VMS, and Unix (4.2BSD) for a yearly license fee of $1,400 for commercial users.

C CODE FOR THE PC

source code, of course

Concurrent C	$45
Coder's Prolog in C	$45
LEX	$25
YACC & PREP	$25
Small-C compiler for 8088	$20
tiny-c interpreter & shell	$20
Xlisp 1.5a & tiny-Prolog	$20
C Tools	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders — No credit cards

Circle **no. 250** on reader service card.

DeSmet C

now with 32-Bit Pointer Option

C88 ..... *still* **$109**

The editors' choice for fast compilation and execution. The **price/performance winner** in all major C benchmarks since 1983. Includes Compiler, Assembler, Binder, Librarian, Execution Profiler and Full Screen Editor. Supports both disk and memory resident Overlays. Contains both 8087 and Software floating point support. Full STDIO library.

Large Case Option **$50**

Makes a great C Compiler even better. Adds 32-Bit Pointers to C88 so you can utilize all of your PC. Groups scalar and static data for fast access. Supports the D88 debugger.

D88 ..... **$50**

Gain most of the benefits of an interpreter while losing none of the run-time speed of the C88 compiler. Display C source and variable contents during execution. Set breakpoints by function name or line number. Examine and set variables by name using C expressions.

order direct from:

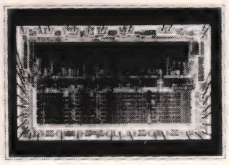
C Ware Corporation
505 W. Olive, Suite 767, Sunnyvale, CA 94086 U.S.A.
(408) 720-9696 — Telex: 358185
We accept VISA, MasterCard & American Express

FROM THE DEVELOPERS OF THE

65816 Microprocessor

The programming handbook you've been waiting for...

Programming the
65816 Microprocessor
Including 6502 and 65C02



Brady David Eyes

- Outlines the programming strengths of the 65816, 6502, and 65C02.
- Includes a review of basic concepts, architecture, and logical operations.

Now available at your local bookstore, or call toll free 1(800)624-0023 to order your copy today. In New Jersey, call 1(800)624-0024.

0-89303-759-3/288 p/$22.95

Brady

Circle **no. 219** on reader service card.

¿C? ¡Sí!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can **use** on **any** machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only $18 for a full year (regularly $28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal!**

Please add $9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. **The C Journal:** InfoPro Systems. UNIX: AT&T Bell Labs.

InfoPro Systems
3108 Route 10
Denville, NJ 07834
(201) 989-0570

MasterCard VISA

Circle **no. 194** on reader service card.

OF INTEREST
(continued from page 123)

White Sciences' Icon Builder software allows the generation of graphics images that can be printed, overlayed on a digitizing tablet surface, and used to augment the limited keyboard space in the construction of icon-oriented user interfaces to application programs. Icon Builder is composed of four software modules: a graphics program, a template editor, a template install program, and an overlay print program. It retails for $79.95.

BMC Software's Data Packer II is a second-generation IMS utility that provides multiple database compression options. Data Packer II reduces DASD space requirements, often by more than 75 percent, thereby reducing the many direct costs affected by DASD needs and high transaction levels. The product is available at $25,000 for a perpetual lease on the first CPU.

RM/COBOL from **Ryan-McFarland Corp.** is a GSA-certified implementation of the ANSI X3.23 74 COBOL standard. It is available under an OEM's custom operating system or under standard systems. RM/COBOL features an indexed file-access method, record- and file-level locking, full arithmetic capability, and interactive screen-handling capabilities.

XTree, Version 2.0, from **Executive Systems** is designed to simplify file and directory handling by providing single keystroke commands to access, delete, rename, view, move, list, or show files within any directory on a floppy and hard disk. Version 2.0 requires an IBM PC or PC/AT with 19K of memory and MS-DOS 2.0 or PC-DOS. The program retails for $49.95.

The Sibec-II, a single-board microcontroller, is available from **Binary Technology.** Sibec-II features the 8052-AH CPU with full floating-point BASIC. The auto baud rate RS-232 connector allows users to connect a terminal and begin programming. The unit is available for $295.

Reference Map

Alsys Inc., 1432 Main St., Waltham, MA 02154; (617) 890-0030. Reader Service Number 16.

Analog & Digital Peripherals Inc., 815 Diana Dr., Troy, OH 45373; (513) 339-2241. Reader Service Number 17.

Applied Data Research LInc., Rte. 206 and Orchard Rd., CN-8, Princeton, NJ L08540-9936; (201) 874-9000. Reader Service Number 18.

Artek Corp., 101 Seaview Dr., Secaucus, NJ 07094; (201) 867-2900. Reader Service Number 19.

Binary Technology Inc., Main St., P.O. Box 67, Meriden, NH 03770; (603) 469-3232. Reader Service Number 20.

BMC Software, P.O. Box 2002, Sugar Land, TX 77478; (713) 240-8800. Reader Service Number 21.

California Software Products Inc., 525 N. Cabrillo Park Dr., Santa Ana, CA 92701; (714) 973-0440. Reader Service Number 22.

CBIS Inc., 2323 Cheshire LBridge Rd., Atlanta, GA L30324; (404) 634-3079. Reader Service Number 23.

Computer Innovations Inc., 980 Shrewsbury Ave., Tinton Falls, NJ 07724; (201) 542-5920. Reader Service Number 24.

Executive Systems Inc., 15300 Ventura Blvd., Ste. 305, Sherman Oaks, CA 91403; (800) 634-5545, in CA (800) 551-5353. Reader Service Number 25.

Flagstaff Engineering, 1120 Kaibab Ln., Flagstaff, AZ 86001; (602) 779-3341. Reader Service Number 26.

Hallock Systems Co. Inc., 267 N. Main St., Herkimer, NY 13350; (315) 866-7125. Reader Service Number 27.

JMI Software Consultants Inc., 904 Sheble Ln., P.O. Box 481, Spring House, PA 19477; (215) 628-0846. Reader Service Number 28.

KnowledgeWare Inc., 2006 Hogvack, Ann Arbor, MI L48105; (313) 971-5363. Reader Service Number 29.

Lamar Micro, 2107 Artesia Blvd., Redondo Beach, CA 90278; (213) 374-1673. Reader Service Number 30.

Lugaru Software Ltd., 5740 Darlington Rd., Pittsburgh, PA 15217; (412) 421-5911. L Reader Service Number 31.

McGraw-Hill Software, Ste. 1350, 8111 LBJ Fwy., Dallas, TX 75251; (214) 437-7422. L Reader Service Number 32.

Microsoft Corp., 16011 N.E. 36th Wy., P.O. Box 97017, Redmond, WA 98073-9717; (206) 882-8080. Reader Service Number 33.

Microtec Research, 3930 LFreedom Cir., Ste. 101, Santa Clara, CA 95054; (800) 551-5554; in CA (408) 733-2919. Reader Service Number 34.

Netron Inc., 99 St. Regis LCrescent North, Downsview, Ont., Canada M3J 1Y9; (416) 636-8333. Reader Service Number 35.

Nial Systems Ltd., P.O. Box 280, Alexandria Bay, NY 13607-0280; (800) 267-0660. Reader Service Number 36.

Oasys, 60 Aberdeen Ave., Cambridge, MA 02138; (617) 491-4180. Reader Service Number 37.

Peachtree Technology Inc., 3120 Crossing Park, Norcross, GA 30071; (404) 662-5158. Reader Service Number 38.

Production Systems Technologies Inc., 642 Gettysburg St., Pittsburgh, PA 15206; (412) 362-3117. Reader Service Number 39.

Quadram Corp., One Quad Wy., Norcross, GA 30093; (404) 923-6666. Reader Service Number 40.

Raima Corp., 12201 S.E. Tenth St., Bellevue, WA 98005; (206) 747-5570. Reader Service Number 41.

Ryan-McFarland Corp., 609 Deep Valley Dr., Rolling Hills Estates, CA 90274; (213) 541-4828. Reader Service Number 42.

Samsung Semiconductor Inc., 5150 Great America Pkwy., Santa Clara, CA 95054; (408) 492-8200. Reader Service Number 43.

Soft Advances, P.O. Box 49473, Austin, TX 78765; (512) 478-4763. Reader Service Number 44.

SoftCraft Inc., P.O. Box 9802, #917, Austin, TX 78766; (512) 346-8380. Reader Service Number 45.

Software Store (The), 706 Chippewa Sq., Marquette, MI 49855; (906) 228-7622. Reader Service Number 46.

STSC Inc., 2115 E. Jefferson St., Rockville, MD 20852; (301) 984-5000. Reader Service Number 47.

Subject, Wills & Co., 800 Enterprise Dr., Oakbrook, IL 60521; (312) 789-0240. Reader Service Number 48.

Tartan Laboratories, 477 Melwood Ave., Pittsburgh, PA 15213; (412) 621-2210. Reader Service Number 49.

TDI Software Inc., 10410 Markison Rd., Dallas, TX 75238; (214) 340-4942. Reader Service Number 50.

Watcom Products, 415 Phillip St., Waterloo, Ont., Canada N2L 3X2; (519) 886-3700. Reader Service Number 51.

White Sciences Inc., P.O. Box 24756, Tempe, AZ 85282; (602) 967-8257. Reader Service Number 52.

—Wendelin Colby

DDJ

Transform Your Programs with CPP—C Preprocessor Plus

Includes ALL features of the standard C preprocessor.
- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Include lines with #if, #ifdef and #ifndef commands.
- Define multiple constants with #enum command.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or * are ignored.)

Fast and flexible
- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

Complete
- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

Price: $95. Call or write today:
Edward K. Ream
1850 Summit Ave., Dept. DD
Madison, WI 53705
(608) 231-2952

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5¼ inch disk). Send a check or money order for $95 ($105 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check is included.

Circle **no. 90** on reader service card.

ATTENTION COBOL USERS

THE COBOL SHOP

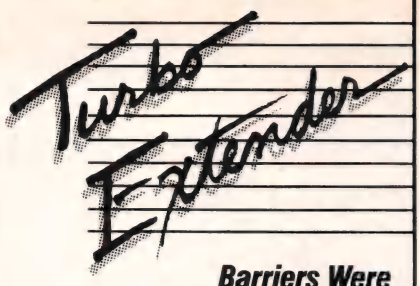
QUALITY PRODUCTS FROM TOP MANUFACTURERS

- The Latest in Compilers and Tools for Dos, Netware, Xenix, Unix, Cics, Ims and More
- Applications Generators
- Relational DBMs, SQL, 4GL
- Forms, Maps, Editors, Debuggers, File Management Systems

Best Products, **Best Prices,** Best Support

Call or Write:
(713) 641-3440
P.O. Box 672
Richmond, TX 77469

Circle **no. 247** on reader service card.



Barriers Were Made To Be Broken

If you're using Turbo Pascal, Turbo Extender is the software you need to break the 64K barrier. Now write programs as long as you need, limited only by available RAM.

LARGE CODE MODEL

Write programs using all 640K without overlays or chaining. Convert existing programs with minimal effort. Offers advanced dynamic binding or standard EXE files.

MAKE FACILITY

Offers separate compilation. No starting from scratch for source code changes.

LARGE DATA ARRAYS

Automatically manages data arrays up to 30 megabytes, as well as arrays stored in expanded memory.

ADDITIONAL TOOLS

Overlay Analyst, Disk Cache, Pascal Encryptor, Shell File Generator, and File Browser.

TURBO EXTENDER

Includes 2 DSDD disks, complete source code, 150 page printed manual. Requires Turbo Pascal 3.0 and PCDOS 2.x or 3.x. Runs on IBM PC/XT/AT and compatibles. Call for MSDOS support. $85 COMPLETE.

ALSO AVAILABLE TURBOPOWER UTILITIES

Bruce Webster, in BYTE Magazine, February 1986, named it Product of the Month, saying "If you own Turbo Pascal, you should own TurboPower Utilities, that's all there is to it."

It offers 9 programs in all, including the Program Structure Analyzer which locates subtle coding problems the compiler fails to catch. It also produces reports that describe your code, providing cross reference and hierarchy diagrams. Also includes Execution Timer, Execution Profiler, Pretty Printer, Command Repeater, Pattern Replacer, Difference Finder, File Finder and Super Directory.

TURBOPOWER UTILITIES

Come compiled, ready to use. Includes manual, reference card, three DSDD disks, complete source code. Requires Turbo Pascal 2.0 or 3.0. $95 with source code; $55 executable only.

GET BOTH TURBO EXTENDER AND TURBOPOWER UTILITIES (Source) $149

MC/VISA CALL TOLL-FREE 7 days a week
(US) 800-538-8157 x830
(CA) 800-672-3470 x830 For PO, COD, Dealers, Info, Brochures – call or write:

TURBO Power™

478 W. Hamilton #196
Campbell, CA 95008
(408) 378-3672
M-F 9AM-5PM PST

INTERNATIONAL REPRESENTATIVES
Switzerland: Software Haus 064-512651
Japan: Southern Pacific Ltd. 045-314-9514
Norway: Polysoft 03-825275
England: Grey Matter Ltd. 0364-53499
Australia: Videogram Communications 02-627-1261

Circle **no. 207** on reader service card.

ADVERTISER INDEX

Reader Sevice No.	Advertiser	Page No.
*	A.T.&T.	11
273	Alsys Inc.	14–15
297	American Micro Technology	35
171	Answer Software	93
121	Arity Corp.	2
216	Atron	26
250	Austin Code Works	123
115	Barrington Systems Inc.	4–5
159	Blaise Computing	19
272	Blast/Communicaitons Res. Grp.	66
161	Borland International	13
285	Brady Computer Books	41
219	Brady Computer Books	124
181	C Users Groups	97
*	C Ware	123
247	COBOL Shop	125
226	Cauzin Systems, Inc.	36–37
81	Cogitate, Inc.	99
237	CompuServe	21
122	CompuView	127
282	Cosmos	44
*	Creative Programming	70
265	D&W Digital	71
258	Desktop A.I.	111
*	Digital Equipment Corp.	C3
87	Digital Research Computers	55
*	DDJ Allen Holub-Shell	78
*	DDJ Back Issues	67
*	DDJ Bound Volumes	74–75
*	DDJ Code Listings	79
*	DDJ C-Products	76–77
*	DDJ Z80 Toolbook	80
89	Ecosoft, Inc.	105
90	Edward K. Ream	125
138	Essential Software	43
93	FairCom	93
*	Gimple Software	39
97	Greenleaf Software	17
132	Harvard Softworks	84
274	Hauppauge Computer Works	C4
*	House ad well	
278	ICD	72
194	InfoPro Systems (Trade)	124
*	Integral Quality	111
79	Interface Technologies	87
186	Lahey Computer Systems	106
101	Lattice, Inc	29
252	Levien Instrument Co.	111
118	Lifeboat	31
257	Logitech, Inc.	C2
223	Manx Software	95
109	Manx Software	68
222	Manx Software	82
108	Manx Software	7
110	Micro Interface Corp.	106
94	MicroPlot	93
105	Microprocessors Unlimited	99
*	Microsoft	1
*	Microsoft	47
126	Microsoft Press	99
125	Microsoft Press	97
*	Microtec Research	121
261	Miken Optical Co.	109
156	Mix Software	69
128	Morgan Computing	109
243	Norton Utilities	85
254	Oasys	34
192	Overland Data Inc.	107
239	PMI	107
76	Personal Tex	94
91	Phoenic Computer Products	119
169	Poor Person Software	67
143	Programmer's Shop	53
141	Programmer's Shop	88–89
129	Programmer's Connection	63,64,65
295	Proto PC, INc.	101
144	Quantum	27
206	Raima Corp.	23
145	Rational Systems	115
120	Relational Memory Systems	97
*	SAS Institute Inc.	51
78	SLR Systems	30
85	Semi Disk Systems	100
83	Soft Advances	85
259	Soft Focus	103
284	SoftLogic Solutions	57
293	SoftLogic Solutions	59
163	SoftLogic Solutions	61
180	Softsmarts, Inc.	45
148	Solution Systems	28
152	Solution Systems	28
153	Solution Systems	114
155	Solution Systems	114
147	Solution Systems	91
298	Sophco	101
164	Spruce Technology Corp.	103
288	Subject, Wills & Co.	40
172	Sunny Hill Software	104
173	TLM Systems	81
175	TLM Systems	85
174	TLM Systems	83
230	TSF	101
245/279	Tech PC	25
207	Turbo Power Software	126
77	UniPress Software	67
291	Visual Age	118
112	Wendin, Inc.	9
116	Wizard Systems	49

*This advertiser prefers to be contacted directly: see ad for phone number.

ADVERTISING SALES OFFICES

Northeast
Shawn Horst (415) 366-3600

Midwest
Michele Beaty (317) 875-8093

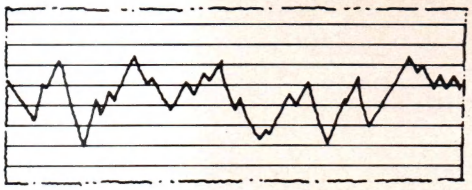
Southeast
Gary George (404) 355-4128

Northern California/Northwest
Lisa Boudreau (415) 366-3600

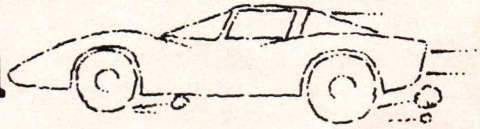
Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600

Advertising Director
Robert Horton (415) 366-3600

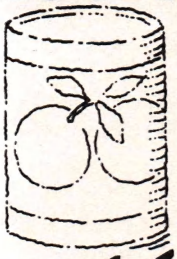
VEDIT® Plus Text Editor



The Navy charts new concepts with it... GM engineers the future with



it... National Can preserves facts with it...GE has bright ideas with it... Here's why you shouldn't be without it.



Every day, VEDIT PLUS helps thousands of programmers, writers and engineers get down to business.

So why do people who could have ANY text editor prefer ours? For a lot of reasons, including:

- CAPACITY—With VEDIT PLUS, file size is never a problem. And virtual disk buffering simplifies editing of even the largest files.
- FLEXIBILITY—VEDIT PLUS lets you edit up to 37 files simultaneously. So you can cut and paste. Edit programs. Edit text. Even perform numerous search/replace functions in several files without user intervention.*
- CUSTOMIZATION—With VEDIT PLUS, you can create your own on-line editing functions with keystroke macros. Develop your own on-line help screens. Determine and revise your own keyboard layout easily.
- SPEED—VEDIT PLUS not only works hard, it works fast. Faster, in fact, than any other text editor on the market.
- EXPERIENCE—Six years ago, CompuView revolutionized the concept of microcomputer text editing. And we've been improving our products and services ever since.

Special Offer: Order a VEDIT PLUS text editor for $225 and we'll include our V-PRINT™ document formatter—a $120 value—absolutely free.

Call CompuView today at 313/996-1299. You'll be in good company.

CompuView®

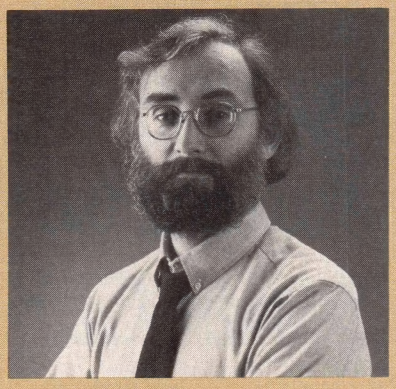
CompuView® Products Inc., 1955 Pauline Boulevard—Suite 300, Ann Arbor, Michigan 48103, TELEX 701821

Available for PC DOS, MS-DOS, CP/M, CP/M-86.

*Free sort, compare, print and main menu macros included; optional 8080-8086 translator or mailmerge, $50 each.

Circle **no. 122** on reader service card.

SWAINE'S FLAMES



Exactly two years ago we published a review of Borland's Turbo Pascal; within the next two months we expect to review Turbo Prolog, assuming we get the product in time (the editor's perennial lament). When I finish this column I intend to drive to Scotts Valley and request a copy in person. That sometimes works.

Column? you ask, flipping through back issues, asking your fickle memory what was on this page before. Certainly not this column with a goofy photo of the editor-in-chief posed like Peter Norton. What are they doing to *DDJ* now?

Glad you asked. This is indeed a new column, written by me, Mike Swaine, editor-in-chief, a column born of the need to have new editor Nick Turner and me flame in parallel (he gets the editorial page) and of the desire to put an opinion column on the last page like many other magazines do.

Oh, fine. Now *DDJ* will start sounding like *InfoWorld*. Ersatz Dvorak, right? Well, no; I hope this will be a *DDJ*-type back-page column, dealing with *DDJ*-type issues from a *DDJ*-type perspective. If the styles of other columnists influence this column, they will be the the columnists who influenced me in my formative years.

I read and enjoyed John Campbell's convention-challenging editorials in *Analog Science Fiction* magazine even after I grew up and learned that they were sophomoric and slightly cracked. I was permanently warped by Martin Gardner's rich, witty, and diligently researched Mathematical Games column in *Scientific American*, and I imitated it in a puzzle column of my own on the back page of *InfoWorld* for a year or so. More recently, I liked Hal Hardenbergh's quirky column-that-ate-the-newsletter in *DTACK Grounded*, but I don't have room here to imitate Hal.

What will this column cover? The usual stuff: significant new software products, books, trends, phenomena. I admire Jon Bentley's Programming Pearls in *Communications of the ACM*, which have been collected into a book also titled *Programming Pearls* (Reading, Mass.: Addison-Wesley, 1986). Bentley has staked out as his domain insight and creativity in programming. Fertile ground, which he tills like a native. I think his is the best new computer book of 1986.

A good column should stimulate its readers to think, not try to think for them. I don't, for example, know the significance of Microsoft's decision not to support .COM files in future versions of DOS (beyond the fact that it's a repudiation of DOS's illegitimate descent from CP/M), but I suspect there may be ramifications that Microsoft hasn't considered.

A good column asks questions, but not just the most obvious ones. Will Borland sell hundreds of thousands of copies of Turbo Prolog? Maybe, but what would it signify if it did? Oddly, despite the fact that Pascal and PROLOG are from different planets, the experience of Turbo Pascal could provide an idea of how Turbo Prolog will be received—breathless reviews in the computer press, reckless spending by under-informed computer owners, confusion over the significance of the PROLOG language, mistaking a good user interface for product depth. Turbo Prolog could be absurdly successful for reasons no less absurd. And yet, the effect on professional software development could turn out to be negligible.

Finally, I hope to tell about the work of an enterprising software developer with a genuinely new idea each month. Take my cousin Corbett, who, having lost his shirt in the software look-alike market when his line of software (called Look and Feel Ware and marketed under the Kalvin Klone label) ran up against some stiff competition, hit upon one of those ideas that leave you speechless with awe.

Corbett's latest line of products is called Tomorrow's Software. Tomorrow's Software does nothing except display so-far-unused icons, shapes, and colors on the screen. Corbett's visionary idea is to stake out new visual metaphors in order to collect royalties from people who will later learn what to do with them. The hall-closet metaphor. The hero-sandwich metaphor. The sheep-entrails metaphor. I think he's onto something. Just yesterday he called to tell me that he'd found a color that had never been put to functional use on the computer screen, and he was applying for a patent on its use. The big question is whether he can sue Steven Spielberg over his use of the color purple.

Yeah, but what's this Borland business? you ask, ignoring the last two paragraphs. If Turbo Prolog may have only negligible significance for software developers, why review it in a software developer's magazine? Well, what I really suspect is that Borland is onto something. I think it's just possible that Turbo Prolog will be significant in the history of software, but it will be so only if Borland is successful in getting it quickly into college classrooms and only if the product is good.

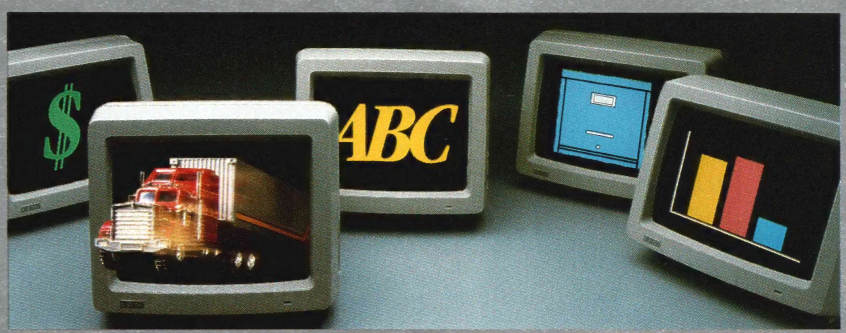
So if you'll excuse me, I'm off to Scotts Valley.

Late news dept.: Dvorak makes **bold** move to *PC Magazine*.

Michael Swaine

Michael Swaine
editor-in-chief

Digital has it now.



The first open-ended multi-user software system that integrates your custom application with a full range of standard business programs.

The A-to-Z™ Integrated System for MicroVAX II™ and MicroPDP-11.™ Now you can integrate any number of your custom multi-user applications with word processing, graphics, spreadsheet, data management, accounting, and thousands of VAX™ and PDP-11 programs. Same menus, commands and files with no need for a system manager.

Tell me about A-to-Z on MicroVAX™ and MicroPDP-II.™ ☐ Reseller ☐ Developer

Name _____ Title _____

Company _____ Phone _____

Address _____

City _____ State _____ Zip _____

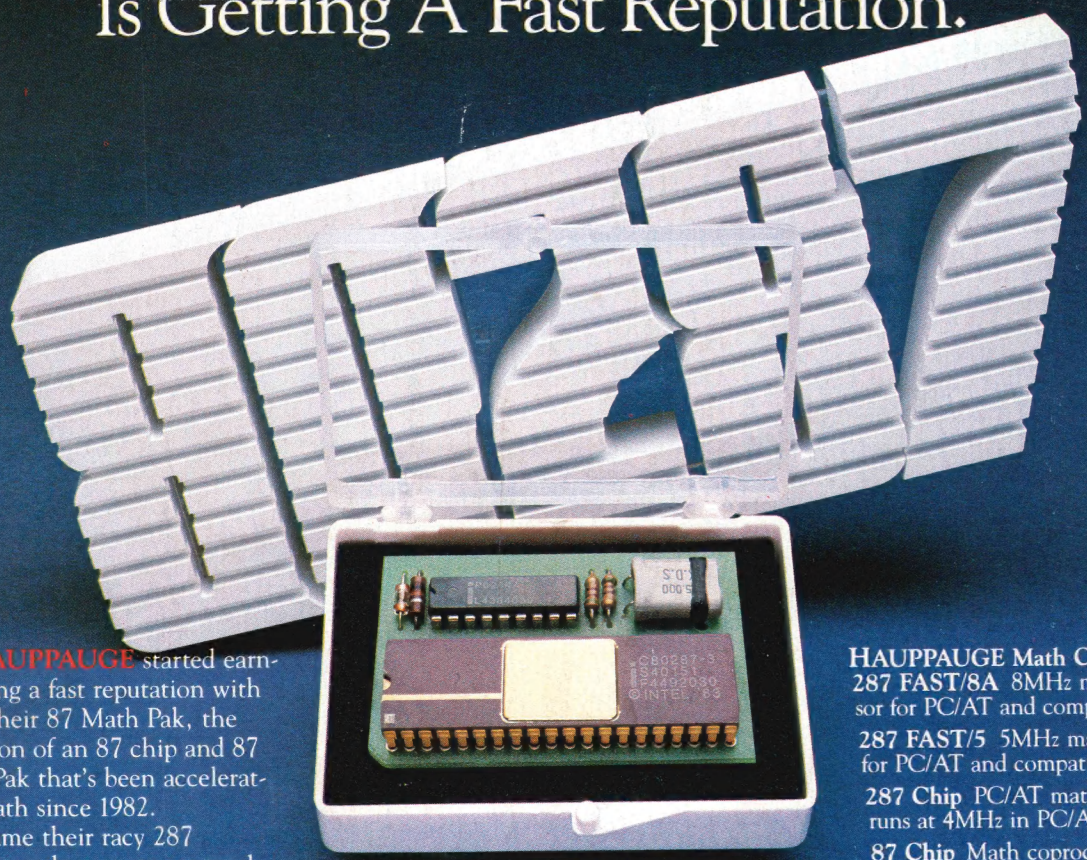
Send to: Digital Equipment Corporation, Inquiry Dept., NR02-1/H3, 444 Whitney Street, Northboro, MA 01532.

digital™

© Digital Equipment Corporation 1986. Digital, the Digital logo, A-to-Z, MicroVAX II, MicroPDP-11, and VAX are trademarks of Digital Equipment Corporation. DD686

HAUPPAUGE

Is Getting A Fast Reputation.



HAUPPAUGE started earning a fast reputation with their 87 Math Pak, the combination of an 87 chip and 87 Software Pak that's been accelerating PC math since 1982.

Next came their racy 287 FAST/5, a math coprocessor module with its own 5MHz clock, speeding up PC/AT math by 25%. (Pictured above.)

Now, Hauppauge Unveils the 287 FAST/8A...
Our newest math coprocessor for the PC/AT, the 287 FAST/8A moves out at 8MHz—doubling the speed of each floating point math operation. The FAST/8A accelerates AutoCad, 1-2-3, Symphony, Turbo Pascal, Framework and more. The FAST/8A also runs in PC/AT compatibles including the Compaq Deskpro 286, Sperry PC/IT and TI Business-Pro computers.

...And the 87 Software Pak Version 6.0
Designed to steal the heart of programmers, the 87 Software Pak supports IBM's BASIC Compiler 1.0 and 2.0, and Microsoft's QuickBASIC, executing math-intensive programs up to 20 times faster! The 87 Software Pak also performs FFT's and Matrix operations. For example, a PC (or PC/XT) with an 87 Chip and 87 Software Pak can perform a 512-point complex FFT in just 1.1 seconds. What's more, a PC/AT with a FAST/8A inverts a 25 by 25 element matrix in under 1 second.

Circle **no. 274** on reader service card.

HAUPPAUGE Math Coprocessors

287 FAST/8A 8MHz math coprocessor for PC/AT and compatibles....$379

287 FAST/5 5MHz math coprocessor for PC/AT and compatibles.....$249

287 Chip PC/AT math coprocessor—runs at 4MHz in PC/AT.....$219

87 Chip Math coprocessor for IBM PC, PC/XT and compatibles $129

87-2 Chip Math coprocessor for 8MHz PC compatibles...$195

HAUPPAUGE Math Coprocessor Paks

87 Math Pak V.6.0 87 chip and math coprocessor software support for IBM BASIC Compiler 1.0, 2.0 and Microsoft's QuickBASIC. Plus, Matrix and FFT support, one year of free updates, complete source code and "8087 Applications and Programming" $279

87 Software Pak V.6.0 Math coprocessor software support as in the 87 Math Pak, but without 87 chip.....$180
With any Hauppauge math coprocessor $150

Recalc+ Math coprocessor support for 1-2-3 version 1A ..$ 95
With any Hauppauge math coprocessor .....$ 49

HFT+ Complete Hayes Fourier Transform Package $125
With any Hauppauge math coprocessor .....$ 79

The 287 FAST/8 Doubles Your PC/AT's Math Speed!
Help your PC/AT get a fast reputation with Hauppauge's new 287 FAST/8A. Call today, or contact your local computer dealer to learn more about Hauppauge's racy product line. And ask for "87 Q & A," our free booklet on math coprocessors.

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway, Suite MSI,
Commack, New York, USA 11725 • 516-360-3827
Available at your local computer dealer

Hauppauge
(Pronounced "Ha-pog")

We acknowledge the following registered trademarks: 1-2-3 and Symphony: Lotus Development Corporation; AutoCad: AutoDesk, Inc.; IBM, PC, PC/AT, PC/XT; Compaq; Deskpro; Sperry; PC/IT; Texas Instruments Business Pro; Ashton-Tate; Framework.